Definição de mecanismos para avaliação de qualidade de produtos de software

Vangrei Fonseca Simão

Orientador:

Prof. Dr. ***José Carlos Maldonado***

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

**USP – São Carlos
fevereiro/2003**

“VERSÃO REVISADA APÓS A DEFESA”

Data da Defesa: 10/09/2001

Visto do Orientador:

À minha família, pelo incentivo e pelos ensinamentos
que foram muito importantes para que eu
conseguisse vencer mais esta etapa.

Agradecimentos

A Deus por ter me oferecido tantas oportunidades e ter me guiado até o fim de mais uma etapa de minha vida.

Ao meu orientador, Prof. Dr. José Carlos Maldonado, pela confiança depositada, pelos valiosos conselhos, pela paciência e amizade.

Aos meus pais, João e Maria (*in memorian*) pelo carinho, dedicação e incentivo.

As minhas irmãs Carla e Carina, meus avôs Manoel, Sinésia (*in memorian*), José (*in memorian*) e Ana.

A Mariângela, pelo carinho e compreensão, principalmente nos momentos mais difíceis passados durante este trabalho.

A todos os professores que, de alguma forma, contribuíram para o meu aprendizado. Em especial a Profa. Dra. Rosely Sanches, Profa. Dra. Solange Rezende e Profa. Dra. Renata Pontin de Mattos Fortes.

A Profa. Dra. Eliana Xavier Linhares de Andrade, Prof. Dr. Odelar Leite Linhares, Prof. Dr. Alagacone Sri Ranga e Prof. Dr. Rildo Ribeiro do Santos pelo incentivo e apoio ao ingresso no programa de mestrado.

Ao Prof. Dr. Fernão Germano e a Profa. Dra. Rosângela Penteado, pelos seminários e discussões realizados sobre Engenharia Reversa.

Aos meus companheiros de república: José Geraldo, Igor e Denilson pela amizade e por todos os momentos inesquecíveis que passamos juntos.

Aos amigos do Grupo de Engenharia de Software, sempre companheiros e prestativos no esclarecimento de dúvidas: Adenilso, André, Andréa, Aline, Auri, Daniela, Djalma, Elisa, Ellen, Emerson, Janaína, Jusane, Luciana, Mayb, Reginaldo, Rejane, Rodrigo, Rosana, Rurik, Sandra, Simone, Tatiana, Thelma, Valéria, Waine, William, Willie.

A todos os amigos conquistados em São Carlos, pelos momentos agradáveis e enriquecedores, que tornaram cada dia, um dia especial: Alexandre, Cláudio, Dinamérico, Enzo, Gedson, Gilson, Kleber, Maxell, Rômulo, Walter.

Aos meus amigos de S. J. Rio Preto: Marcelo, Ricardo, José Eduardo, Adriano, Paulo, Edson, Fábio, Adinan.

Ao pessoal da seção de Pós: Laura, Marília e Beth, e a todos os funcionários do ICMC, por toda simpatia e atenção dispensada.

A todos aqueles que direta ou indiretamente colaboraram para o desenvolvimento e conclusão deste trabalho.

Ao CNPq, pelo apoio financeiro.

Resumo

Este trabalho apresenta a modelagem de um processo de engenharia reversa e a definição de um modelo de qualidade associado, visando à obter indicativos da qualidade do produto atual e dos artefatos produzidos durante este processo. Para a realização da modelagem foi utilizado o método FUSION-RE/I, o qual tem como objetivo auxiliar na atividade de manutenção, recuperando visões funcionais e estruturais de um sistema, e a abordagem SQUID (método e ferramenta), que é uma abordagem de desenvolvimento de software que visa a garantir e controlar a qualidade de um software através de medições de artefatos intermediários produzidos pelo processo de desenvolvimento ao qual o software está submetido. A abordagem SQUID envolve tanto a visão de desenvolvimento (modelo de desenvolvimento) quanto a visão de qualidade (modelo de qualidade), que devem estar associadas em função das características de um particular projeto.

Abstract

This work presents the modeling of a reverse engineering process and the definition of the associated quality model, aiming at obtaining evidence of the quality of artifacts and the products produced during this process. For the accomplishment of the modeling, the FUSION-RE/I method and the SQUID approach were used. The FUSION-RE/I is a reverse engineering method that leads to recover functional and structural visions of a system. The SQUID approach (method and tool) aims at controlling and guaranteeing the quality of a software through measurements of the intermediate artifacts produced during the development process to which the software is submitted. The SQUID approach involves two different visions: the development vision (development model) and the quality vision (quality model). These visions should be associated in accordance to the characteristics of a particular project.

SUMÁRIO

Lista de Figuras.....	iv
Lista de Tabelas.....	v
Capítulo 1 – Introdução.....	1
1.1 – Contexto.....	1
1.2 – Motivação e Relevância.....	1
1.3 – Objetivo.....	3
1.4 – Organização do Trabalho.....	3
Capítulo 2 – Qualidade de Software.....	5
2.1 – Considerações Iniciais.....	5
2.2 – Visão Geral de Qualidade.....	5
2.3 – Qualidade de Software.....	6
2.4 – Modelos para Avaliação de Produto de Software.....	8
2.4.1 – O Modelo de McCall.....	8
2.4.2 – A Norma NBR 13596.....	10
2.4.2.1 – As Características de Qualidade da NBR 13596.....	10
2.4.3 – Norma NBR ISO/IEC 12119.....	13
2.4.3.1 – Requisitos de Qualidade.....	14
2.4.3.2 – Instruções para Teste.....	16
2.4.4 – Norma ISO/IEC 14598-5.....	17
2.5 – Modelos de Avaliação e Melhoria de Processo de Software.....	19
2.5.1 – Capability Maturity Model (CMM).....	19
2.5.1.1 – Nível 1: Inicial.....	20
2.5.1.2 – Nível 2: Repetitivo.....	21
2.5.1.3 – Nível 3: Definido.....	21
2.5.1.4 – Nível 4: Gerenciado.....	22
2.5.1.5 – Nível 5: Otimizado.....	22
2.5.2 – Norma NBR ISO/IEC 12207.....	23
2.5.3 – ISO/IEC TR 15504.....	26
2.5.3.1 – Parte 1: Conceitos e Guia Introdutório.....	28
2.5.3.2 – Parte 2: Descrição de um Modelo Idealizado de Processo.....	28
2.5.3.3 – Parte 3: Classificação dos Processos.....	29
2.5.3.4 – Parte 4: Guia de Orientação para Condução da Avaliação.....	29
2.5.3.5 – Parte 5: Diretrizes para Elaboração de um Instrumento de Avaliação.....	29
2.5.3.6 – Parte 6: Qualificação e Treinamento de Avaliadores.....	30
2.5.3.7 – Parte 7: Guia de Orientação para Melhoria de Processo.....	30
2.5.3.8 – Parte 8: Guia de Orientação para Determinação da Capacidade de Processo.....	30
2.5.3.9 – Parte 9: Vocabulário.....	30
2.5.4 – ISO 9000-3.....	31
2.5.4.1 – Estrutura.....	32
2.5.4.2 – Atividades do Ciclo de Vida.....	33
2.5.4.3 – Atividades de Suporte.....	34
2.6 – A Abordagem Goal/Question/Metric (GQM).....	36
2.6.1 – Visão Geral da Abordagem GQM.....	36

2.6.2 – Estabelecer os Objetivos da Coleta de Dados	38
2.6.3 – Desenvolver uma Lista de Questões de Interesse	39
2.6.4 – Estabelecer as Métricas	39
2.6.5 – Projetar e Testar o Questionário de Coleta de Dados	40
2.6.6 – Coletar e Validar os Dados	40
2.6.7 – Analisar os Dados	41
2.7 – A Qualidade no Setor de Software Brasileiro	41
2.8 – Considerações Finais	45
Capítulo 3 – Engenharia Reversa de Sistemas	46
3.1 – Considerações Iniciais	46
3.2 – Níveis de Abstração no Ciclo de Vida	46
3.3 – Definições de Engenharia Reversa	47
3.4 – Categorias de Métodos de Engenharia Reversa	49
3.5 – Método de Engenharia Reversa FUSION-RE/I	52
3.5.1 – Processo de Engenharia Reversa FUSION-RE/I Segundo a Notação UML	58
3.5.1.1 – A Linguagem de Modelagem Unificada (UML)	59
3.5.1.2 – Método FUSION-RE/I Segundo a Notação UML	61
3.6 – Considerações Finais	62
Capítulo 4 – Abordagem SQUID	63
4.1 – Considerações Iniciais	63
4.2 – A Abordagem SQUID para Qualidade	63
4.3 – O Modelo de Dados SQUID (SQUID Data Model)	65
4.3.1 – Representando o Processo de Desenvolvimento	66
4.3.2 – Representando o Modelo de Qualidade	66
4.3.3 – Representando Medidas	68
4.3.4 – Integrando Componentes do Modelo de Dados SQUID	68
4.4 – O Processo de Qualidade SQUID (SQUID Quality Process)	69
4.4.1 – Especificação da Qualidade	69
4.4.2 – Planejamento e Controle da Qualidade	71
4.4.3 – Avaliação da Qualidade	72
4.5 – Considerações Finais	73
Capítulo 5 – Processo de Engenharia Reversa Segundo a Abordagem SQUID	74
5.1 – Considerações Iniciais	74
5.2 – Modelo de Desenvolvimento do Processo de Engenharia Reversa	74
5.3 – Modelo de Qualidade do Processo de Engenharia Reversa	78
5.4 – Seleção de Métricas	79
5.4.1 – Métricas Externas	81
5.4.2 – Métricas Internas	83
5.5 – Integração dos Modelos de Qualidade e de Desenvolvimento	89
5.6 – Considerações Finais	93
Capítulo 6 – Instanciando a Modelagem Proposta na Ferramenta SQUID	94
6.1 – Considerações Iniciais	94
6.2 – Configurando a Ferramenta SQUID	94
6.3 – Especificação da Qualidade	98
6.4 – Avaliando a Viabilidade dos Requisitos de Qualidade	100
6.5 – Planejamento da Qualidade	100
6.6 – Medidas Atuais	103

6.7 – Controle da Qualidade.....	105
6.8 – Avaliação da Qualidade.....	106
6.9 – Considerações Finais.....	107
Capítulo 7 – Conclusões e Trabalhos Futuros.....	108
7.1 – Conclusões.....	108
7.2 – Trabalhos Futuros.....	110
Referências Bibliográficas.....	111
Apêndices.....	120
Apêndice A – Relatório SQUID do Modelo de Desenvolvimento Proposto.....	121
Apêndice B – Relatório SQUID do Modelo de Qualidade Proposto.....	122
Apêndice C – Relatório SQUID da Integração dos Modelos de Qualidade e de Desenvolvimento....	124
Apêndice D – Relatório SQUID das Unidades de Medidas Utilizadas.....	129
Apêndice E – Telas de Recursos Estatísticos Disponíveis na Ferramenta SQUID.....	131

LISTA DE FIGURAS

Figura 2.1 – Fatores de Qualidade de Software de McCall [McCall 77]	9
Figura 2.2 – Estrutura da Norma NBR ISO/IEC 12119 [NBR ISO/IEC 12119]	13
Figura 2.3 – Processo de Avaliação [ISO/IEC 14598-5]	18
Figura 2.4 – Processos de Ciclo de Vida do Software [Rocha 01]	24
Figura 2.5 – Avaliação de Processo de Software – ISO/IEC TR 15504 [ISO/IEC TR 15504]	27
Figura 2.6 – Partes Componentes do ISO/IEC TR 15504 [ISO/IEC TR 15504]	28
Figura 2.7 – Partes Componentes da ISO 9000-3 [Rosa 97]	32
Figura 2.8 – Estrutura Hierárquica do Método GQM [Basili 95]	37
Figura 2.9 – Passos do Método GQM	37
Figura 2.10 – Coordenadas da Meta do Método GQM [Basili 95]	38
Figura 3.1 – Níveis e Graus de Abstração no Ciclo de Vida [Costa 97]	47
Figura 3.2 – Visões das Categorias de Engenharia Reversa [Costa 97]	50
Figura 3.3 – Um Processo de Engenharia Reversa [Rosenberg 00]	52
Figura 3.4 – Síntese do Método de Engenharia Reversa FUSION-RE/I [Costa 97]	53
Figura 3.5 – Notação para o Modelo de Objetos segundo o Método Fusion [Coleman 96]	56
Figura 4.1 – Fases do Processo de Qualidade – SQUID [Tsukumo 96]	64
Figura 4.2 – Representando o Processo de Desenvolvimento [SQUID 98]	66
Figura 4.3 – Representando o Modelo de Qualidade [SQUID 98]	67
Figura 4.4 – Representando Atributos e o Modelo de Qualidade na Ferramenta SQUID	67
Figura 4.5 – Representando Medidas [SQUID 98]	68
Figura 4.6 – Representação Simplificada do Modelo de Dados SQUID [Bøegh 99]	69
Figura 4.7 – Especificando os Requisitos de Qualidade (Visão Externa)	70
Figura 4.8 – Planejamento da Qualidade (Visão Interna)	71
Figura 4.9 – Identificando Componentes Anômalos	72
Figura 4.10 – Atividades do Processo de Qualidade e Relações com a Base de Dados SQUID [Bøegh 99]	73
Figura 5.1 – Modelo de Qualidade Proposto	79
Figura 5.2 – Tipos de Métricas do Modelo de Qualidade	81
Figura 6.1 – Tela Inicial da Ferramenta SQUID (Configuração da Ferramenta)	94
Figura 6.2 – Introduzindo o Modelo de Desenvolvimento	95
Figura 6.3 – Introduzindo Atributos de Qualidade	95
Figura 6.4 – Criando Unidades de Medida	96
Figura 6.5 – Representado o Modelo de Qualidade na Ferramenta SQUID	96
Figura 6.6 – Integração dos Modelos de Qualidade e de Desenvolvimento	97
Figura 6.7 – Criando um Novo Projeto	98
Figura 6.8 – Instanciando Subsistemas (Product Portions)	99
Figura 6.9 – Definindo Requisitos de Qualidade	100
Figura 6.10 – Definindo a Visão Interna de um Subsistema	101
Figura 6.11 – Criando Instâncias de Objetos de Projeto para um Determinado Subsistema	102
Figura 6.12 – Atribuindo Valores Objetivos aos Atributos Internos	103
Figura 6.13 – Atribuindo Valores Atuais aos Atributos Externos	104
Figura 6.14 – Atribuindo Valores Atuais aos Atributos Internos	105
Figura 6.15 – Detecção de Componentes Anômalos	106
Figura 6.16 – Avaliação da Qualidade (Não Implementado)	107

LISTA DE TABELAS

Tabela 2.1 – Definição dos Fatores de Qualidade segundo McCall.....	9
Tabela 2.2 – Subcaracterísticas da Funcionalidade [NBR 13596]	11
Tabela 2.3 – Subcaracterísticas da Confiabilidade [NBR 13596]	11
Tabela 2.4 – Subcaracterísticas da Usabilidade [NBR 13596]	11
Tabela 2.5 – Subcaracterísticas da Eficiência [NBR 13596]	12
Tabela 2.6 – Subcaracterísticas da Manutenibilidade [NBR 13596]	12
Tabela 2.7 – Subcaracterísticas da Portabilidade [NBR 13596]	13
Tabela 2.8 – Requisitos de Qualidade para a Descrição de Produto [NBR ISO/IEC 12119]	14
Tabela 2.9 – Requisitos de Qualidade para a Documentação de Usuário [NBR ISO/IEC 12119]	15
Tabela 2.10 – Requisitos de Qualidade para Programas e Dados [NBR ISO/IEC 12119]	16
Tabela 2.11 – Instruções para Teste [NBR ISO/IEC 12119]	16
Tabela 2.12 – Níveis e Áreas-Chave de Processo do Modelo CMM.....	20
Tabela 2.13 – ISO 9000-3 (Estrutura) [ISO 9000-3]	32
Tabela 2.14 – ISO 9000-3 (Atividades do Ciclo de Vida) [ISO 9000-3]	33
Tabela 2.15 – ISO 9000-3 (Atividades de Suporte) [ISO 9000-3]	34
Tabela 2.16 – Porte das Empresas, segundo Comercialização Bruta Anual Proveniente de Software [MCT 00]....	42
Tabela 2.17 – Elaboração de Planos Estratégicos, Planos de Negócios ou Planos de Metas [MCT 00] ..	42
Tabela 2.18 – Inclusão de Metas ou Diretrizes para Qualidade nos Planos [MCT 00]	43
Tabela 2.19 – Conhecimento de Normas para Qualidade [MCT 00]	43
Tabela 2.20 – Avaliação de Produtos Baseados nas Normas de Qualidade de Produto de Software [MCT 00]	43
Tabela 2.21 – Conhecimento do modelo CMM e do relatório técnico ISO/IEC TR 15504 [MCT 00]	44
Tabela 3.1 – Operadores das Expressões Regulares do Modelo de Ciclo de Vida [Coleman 96]	55
Tabela 3.2 – Notação do Modelo de Operações [Coleman 96]	55
Tabela 3.3 – Notação do Quadro de Chamadas de Procedimentos [Costa 97]	57
Tabela 3.4 – Quadro Índice dos Procedimentos [Costa 97]	57
Tabela 3.5 – Quadro Operações/Procedimentos de Implementação [Costa 97]	58
Tabela 3.6 – Método FUSION-RE/I x UML [Feltrim 99a]	61
Tabela 5.1 – Pontos de Revisão	75
Tabela 5.2 – Atividades de Desenvolvimento.....	76
Tabela 5.3 – Produtos Gerados pelas Atividades de Desenvolvimento	77
Tabela 5.4 – Atributos Externos	81
Tabela 5.5 – Descrição das Características e Subcaracterísticas Externas	82
Tabela 5.6 – Visão Externa da Qualidade	82
Tabela 5.7 – Atributos Internos	83
Tabela 5.8 – Descrição das Características e Subcaracterísticas Internas	85
Tabela 5.9 – Visão Interna da Qualidade	86
Tabela 5.10 – Definição de Unidades de Medida	87
Tabela 5.11 – Associação de Unidades de Medida aos Atributos Externos	88
Tabela 5.12 – Integração dos Modelos (Pontos de Revisão)	89
Tabela 5.13 – Integração dos Modelos (Atividades de Desenvolvimento)	90
Tabela 5.14 – Integração dos Modelos (Produtos)	90

Capítulo 1 – Introdução

1.1 – Contexto

O desenvolvimento de software exige, cada vez mais, um processo sistemático e controlado. O mercado consumidor tem se tornado mais rigoroso na escolha de seus produtos e a qualidade dos mesmos passou a ser um fator crítico para a garantia do sucesso. Ao mesmo tempo, desenvolvedores de software sentem a necessidade de melhorar a qualidade de seus produtos para manterem-se competitivos no mercado [Oliveira 95].

A própria evolução histórica do desenvolvimento de software já previa uma época em que a simples existência do software não satisfaria o mercado, que se tornaria mais exigente no sentido não apenas de produção ou aquisição de produtos de software, e sim de produção e aquisição de software de qualidade [Basili 91].

O momento atual e os anos que seguem podem certamente ser chamados de *Era da Qualidade*. Com a maior consolidação do software no mercado, maior é a exigência por garantia de qualidade por parte dos usuários. Aqueles fornecedores que não estiverem aptos a fornecer qualidade em seus produtos poderão ser excluídos do mercado. Neste contexto, o uso de uma abordagem de desenvolvimento de software que auxilie no gerenciamento e controle da qualidade, e de métodos e técnicas de engenharia reversa que apóiem as atividades de manutenção, documentação e reengenharia de sistemas existentes, desempenha um papel fundamental na elaboração/evolução de produtos com melhor qualidade.

1.2 – Motivação e Relevância

Pesquisas realizadas recentemente no Brasil indicam que a maioria das empresas ainda não possui uma cultura voltada para a questão da qualidade (sistema de qualidade, avaliação da qualidade dos produtos, melhoramento de processos, coleta de métricas). Porém percebe-se uma evolução ao se comparar os resultados obtidos através de pesquisas feitas anteriormente e em 1999 [MCT 00][Weber 01].

O incentivo que o governo brasileiro vem dando à exportação de software, através do programa SOFTEX 2000 (Programa Nacional de Software para Exportação), tem promovido um aumento do número de empresas que desenvolvem software. Neste ponto, a qualidade de software destaca-se como um fator de diferenciação decisivo.

Atualmente, encontram-se disponíveis diversos modelos e padrões de qualidade de produto [NBR 13596][NBR ISO/IEC 12119][ISO/IEC 14598-5] e de processos de software [CMM][ISO/IEC TR 15504][NBR ISO/IEC 12207][ISO 9000-3]. É necessária a familiarização e adequação desses modelos de qualidade ao ambiente das empresas para que os mesmos possam ser adotados de um modo viável e prático pelos profissionais responsáveis da área, através de um planejamento e controle de seu uso.

Muitas empresas de desenvolvimento de software estão se adequando às tendências tecnológicas, como a adoção da abordagem orientada a objetos em seus projetos. Em muitas situações faz-se necessário também a introdução de métodos de engenharia reversa que auxiliem na recuperação da documentação atualizada do sistema atual, que pode ser utilizada para melhorar a manutenibilidade e/ou para apoiar a realização de processos de reengenharia. Atualmente existem diversas abordagens de engenharia reversa orientada a objetos [Penteado 96][Costa 97][Cimitile 99]. A abordagem FUSION-RE/I (Fusion – Reverse Engineering / Interface) [Costa 97] é uma delas, e gera documentação de análise orientada a objetos de sistemas desenvolvidos originalmente com orientação a procedimentos, apoiando assim sua transição. Poucas iniciativas tratam a engenharia reversa como um processo (planejamento, controle, métricas) [Sneed 95][Bennett 98][Systa 00]. Este trabalho procura contribuir nesta perspectiva.

Este trabalho de mestrado está inserido dentro do projeto SQUAD (Software Quality Across Different regions) o qual é uma ação conjunta entre entidades européias e sul-americanas, entre eles o ICMC-USP. O Projeto SQUAD visa à coordenação do esforço de pesquisa na área de qualidade de software, contribuindo para a eficiência na atividade de pesquisa e na disseminação e exploração de seus resultados. A ação conjunta SQUAD envolve entidades que têm projetos de pesquisas em andamento na área de qualidade de software e padronização. As atividades e objetivos desses projetos são fortemente relacionados, de forma que um efeito sinérgico pode ser obtido com um mínimo de coordenação e troca de resultados.

Um dos produtos disponíveis utilizados no projeto SQUAD é a ferramenta SQUID (Software Quality In Development), a qual foi financiada pelo programa ESPRIT (European Strategic Programme for Research and Development in Information Technology) da Comissão Européia, e apóia a aplicação de um método desenvolvido para garantia e controle da qualidade (abordagem SQUID). O principal objetivo da ferramenta SQUID é permitir a uma organização de desenvolvimento de software:

- Planejar e controlar a qualidade dos produtos durante o desenvolvimento pelo uso de medições;
- Aprender de uma maneira aprofundada e mais sistemática sobre o processo e produto de software e ter um *feedback* apropriado de experiências atuais auxiliando o planejamento e desenvolvimento de projetos futuros;

- Avaliar a qualidade do produto final.

1.3 – Objetivo

Este trabalho tem por objetivo, com base em modelos e padrões de qualidade de produto e processos de software, abstrair um processo de engenharia reversa e definir um modelo de qualidade associado a esse processo, identificando e desenvolvendo mecanismos para apoiar empresas de software (no contexto brasileiro) na evolução de seus produtos de software, com qualidade e na perspectiva tecnológica atual. Utilizar-se-á a experiência dos parceiros europeus, concretizada na ferramenta SQUID, para o desenvolvimento deste trabalho.

1.4 – Organização do Trabalho

Para os propósitos descritos, este trabalho está dividido em 7 capítulos. Neste capítulo são descritos o contexto no qual este trabalho está inserido, a motivação para realizá-lo e o objetivo a ser atingido.

O Capítulo 2 refere-se especificamente ao tema Qualidade, onde é apresentada uma visão geral do que é qualidade, qualidade de software, descreve o conceito de qualidade de produto e de processo de software, apresenta modelos para avaliação de produtos de software e modelos de avaliação e melhoria de processo de software, descreve a abordagem Goal/Question/Metric (GQM) que é um mecanismo simples e eficaz para o estabelecimento de métricas, além de um panorama da qualidade no setor de software brasileiro.

O Capítulo 3 apresenta os conceitos relacionados à tecnologia de engenharia reversa e o método de engenharia reversa orientado a objetos FUSION-RE/I.

No Capítulo 4 é apresentada a abordagem SQUID (Software Quality in Development) a qual será utilizada no escopo desse trabalho. A ferramenta SQUID é por sua vez descrita a fim de aspectos de modelagem.

No Capítulo 5 é apresentada a modelagem do método FUSION-RE/I e definido um modelo de qualidade associado segundo a abordagem SQUID.

No Capítulo 6 é descrito como a modelagem realizada no capítulo anterior é introduzida na ferramenta SQUID e como instanciar um projeto de engenharia reversa utilizando a modelagem proposta.

Finalmente, no Capítulo 7 são apresentadas as conclusões do trabalho e propostas algumas linhas de pesquisa que possam vir a ser desenvolvidas.

Nos apêndice são apresentados:

- Apêndice A – Relatório SQUID do modelo de desenvolvimento proposto;
- Apêndice B – Relatório SQUID do modelo de qualidade proposto;
- Apêndice C – Relatório SQUID da integração dos modelos de qualidade e de desenvolvimento;
- Apêndice D – Relatório SQUID das unidades de medidas utilizadas;
- Apêndice E – Telas de recursos estatísticos disponíveis na ferramenta SQUID.

Capítulo 2 – Qualidade de Software

2.1 – Considerações Iniciais

Não há dúvidas de que a qualidade se tornou um fator competitivo de grande importância, que pode determinar a hegemonia, a simples sobrevivência ou o desaparecimento de uma organização.

A indústria de software, no entanto, ainda busca caminhos para atingir o ritmo de melhorias significativas visto na manufatura e, mais ainda, os resultados que são exigidos hoje em termos de qualidade e produtividade.

Uma visão geral do conceito de qualidade é apresentada na seção a seguir. A definição de qualidade de software e conceitos relacionados à qualidade de produto e processo de software são apresentados na seção subsequente. As demais seções apresentam alguns modelos para avaliação de produtos de software, modelos de avaliação e melhoria de processos de software, a abordagem GQM que auxilia na identificação e estabelecimento de métricas e um panorama da qualidade no setor de software brasileiro. A abordagem SQUID que envolve tanto a visão de produto como de processo de software será descrita no Capítulo 4.

2.2 – Visão Geral de Qualidade

Na literatura vários autores definiram qualidade sob abordagens diferentes. Segundo Rothery [Rothery 93], qualidade significa "adequação ao uso conforme as exigências feitas". Isto implica dizer que o produto deve executar de maneira apropriada à função para a qual ele foi designado.

Crosby [Crosby 79] define qualidade como "conformidade aos requisitos". Esses devem ser, portanto, claramente definidos para que não sejam mal interpretados. No caso de software, ao longo do seu processo de desenvolvimento, são realizadas medições para verificar a adequação aos requisitos. As não conformidades encontradas são consideradas como defeitos e portanto, como falta de qualidade.

De acordo com Juran [Juran 70], a definição de qualidade está ligada à "adequação ao uso", o que significa levar em consideração os requisitos e as expectativas do cliente. Visto que cada cliente usa o mesmo produto de maneiras diferentes, ele deve possuir a capacidade de atender a cada uma das diversas expectativas de maneira conveniente e satisfatória. Essa capacidade existente no produto compõe o que chamamos de características de qualidade.

Ainda, de acordo com a Norma NBR 13596, "qualidade é a totalidade das características de uma entidade, que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas" [NBR 13596].

2.3 – Qualidade de Software

De um modo geral, qualidade de software pode ser definida como um conjunto de atributos de software que devem ser satisfeitos de modo que o software atenda às necessidades dos usuários. A determinação dos atributos relevantes para cada software varia em função do domínio da aplicação, das tecnologias utilizadas, das características específicas do projeto e das necessidades do usuário e da organização [Kan 95].

Diante disto, existem duas visões de qualidade de software, uma dos usuários e outra dos desenvolvedores, mas ambas consideram que o software não pode falhar. O usuário avalia o software sem conhecer seus aspectos internos, está apenas interessado na facilidade de uso, no desempenho, na confiabilidade dos resultados obtidos e também no preço do software. Os desenvolvedores avaliam aspectos internos como taxa de defeitos, confiabilidade, facilidade de manutenção e também aspectos de conformidade em relação aos requisitos dos clientes [NBR 13596].

Tanto do ponto de vista do usuário quanto do desenvolvedor, os requisitos de software formam a base a partir da qual a qualidade é avaliada. Pode-se afirmar com segurança que a falta de conformidade aos requisitos é falta de qualidade. Existem tipos de requisitos explícitos e implícitos. Se o software atende aos requisitos explícitos, mas não está de acordo com os requisitos implícitos, então a qualidade é duvidosa.

Além dos requisitos, existem padrões específicos que definem os critérios de desenvolvimento que servem como diretrizes para o desenvolvimento do software que está sendo produzido. Se os critérios não são seguidos, o resultado é falta de qualidade [Pressman 00].

Sobre a definição de Qualidade de Software, a abordagem SQUID está em consonância com os princípios de qualidade definida pela Norma NBR 13596. De acordo com a Norma NBR 13596, "Qualidade de Software é a totalidade das características de um produto de software, que lhe conferem a capacidade de satisfazer necessidades explícitas e implícitas" [NBR 13596].

Por outro lado, Kan [Kan 95] define Qualidade de Software de uma maneira extremamente objetiva, explicitando somente a "conformidade aos requisitos do software".

Pressman [Pressman 00] por sua vez, coloca uma definição mais abrangente, na qual "Qualidade de Software é a conformidade aos requisitos de desempenho e de funcionalidade que foram

explicitamente definidos, aos padrões de desenvolvimento explicitamente documentados, e às características implícitas que são esperadas de todo software profissionalmente desenvolvido".

Qualidade de Produto de Software

Qualidade de produto de software refere-se a um conjunto de características que, quando atingidas, contribuem para se obter os objetivos de qualidade do software. Estas características devem estar relacionadas aos requisitos de qualidade estabelecidos para o produto.

Requisitos de qualidade para produtos de software, entretanto, são difíceis de definir devido a vários fatores. Áreas de aplicação diferentes têm requisitos de qualidade diferentes. Além disso, diferentes projetos numa mesma área de aplicação podem apresentar requisitos de qualidade distintos. Ainda, pode haver conflitos entre requisitos dentro de uma mesma área de aplicação uma vez que atingir um requisito em maior grau pode acarretar numa perda em outro requisito [Oliveira 95].

Para auxiliar nesta definição existem normas e métodos que buscam estabelecer características de qualidade desejáveis para um produto de software. Alguns modelos são mais específicos enquanto outros são mais genéricos, no entanto, todos eles mantêm um mesmo foco: definir atributos que contribuam para a utilização do software.

Raramente qualidade pode ser incorporada ao produto final após o processo de desenvolvimento ter terminado. Por essa razão, a qualidade do produto de software é um objetivo do processo de desenvolvimento. Assim, ao desenvolver-se um produto, deve-se ter previamente estabelecidas, como perspectiva, as características de qualidade que se desejam alcançar. Se o processo de desenvolvimento de software considerar as características de qualidade, pode-se dizer que o produto final deverá apresentar essas características [Kan 95].

Qualidade de Processo de Software

Entende-se por processo de software um conjunto de ferramentas, métodos e práticas usadas para produzir um software, ou seja, um processo de software é representado por um conjunto de atividades, metas, transformações e eventos que encapsulam estratégias para o cumprimento da evolução de software [Pressman 00].

Como as exigências por qualidade têm sido cada vez maiores, tornou-se essencial ter-se um controle sobre o processo de desenvolvimento de software de modo que este não se torne um fator inibidor de qualidade e produtividade. O objetivo de gerenciar o processo de software é gerar produtos de acordo com o planejado e simultaneamente melhorar a capacidade da organização em produzir software com maior qualidade [Humphrey 89]. Considerar o processo de desenvolvimento uma atividade que deve ser controlada, medida e melhorada é um passo

importante para torná-lo realmente efetivo. Para tanto é necessário que se tenha controle do relacionamento entre as atividades, as ferramentas, o pessoal, os métodos e principalmente as características do produto de software, que é o objetivo final de todo o processo.

No entanto, definir e gerenciar todo um processo de software de modo a atingir estes objetivos é uma tarefa complexa e minuciosa visto a intensa dinâmica que o rege. Deste modo, torna-se necessário adotar-se um modelo de processo de software que, a partir de metodologias de modelagem adequadas, possa ser usado para desenvolver descrições mais precisas e formais das atividades de ciclo de vida do software [Humphrey 89].

2.4 – Modelos para Avaliação de Produto de Software

A preocupação com a qualidade de um produto de software é cada vez maior em nossos dias. Para que se possa produzir um software de qualidade, contudo, é necessário identificar que atributos determinam a qualidade desejada, estabelecendo seu universo de abrangência [Belchior 92].

Ao longo dos anos, diversos modelos para avaliação da qualidade de software foram propostos, procurando identificar características de qualidade gerais e procedimentos para se determinar a qualidade.

Nesta seção serão abordados modelos para avaliação de produtos de software. São descritos os modelos de McCall [McCall 77], o qual é um dos precursores dos demais modelos usados hoje em dia (entre eles a Norma NBR 13596), a Norma NBR 13596, a Norma NBR ISO/IEC 12119 e a Norma ISO/IEC 14598-5.

2.4.1 – O Modelo de McCall

McCall e outros pesquisadores [McCall 77] propuseram uma categorização dos fatores que afetam a qualidade do software. Estes fatores de qualidade, mostrados na Figura 2.1, focalizam três importantes aspectos do produto de software: suas *características operacionais*, sua *habilidade de suportar mudanças* e sua *adaptabilidade diante de novos ambientes*. As definições dos fatores de qualidade se encontram na Tabela 2.1.

Figura 2.1 – Fatores de Qualidade de Software de McCall [McCall 77]

Tabela 2.1 – Definição dos Fatores de Qualidade segundo McCall

Características operacionais	<i>Corretitude</i>	Medida do quanto o programa satisfaz suas especificações e do quanto cumpre os objetivos visados pelo cliente.
	<i>Confiabilidade</i>	Medida do quanto o programa pode desempenhar as funções esperadas com uma certa precisão.
	<i>Eficiência</i>	O total de recursos computacionais e de código requerido para o programa desempenhar suas funções.
	<i>Integridade</i>	A extensão do quanto o acesso ao software ou a seus dados por pessoas não autorizadas pode ser controlado.
	<i>Usabilidade</i>	O esforço requerido para aprender, operar, preparar dados de entrada, e interpretar os dados de saída do programa.
Habilidade de ser alterado	<i>Manutenibilidade</i>	O esforço requerido para localizar e corrigir erros no programa.
	<i>Flexibilidade</i>	O esforço necessário para modificar um programa operacional.
	<i>Testabilidade</i>	O esforço requerido para testar um programa para garantir que ele desempenhe a função pretendida.
Adaptabilidade a novos ambientes	<i>Portabilidade</i>	O esforço requerido para transferir o programa de um hardware e/ou ambiente de software para outro.
	<i>Reusabilidade</i>	A extensão com que o programa (ou partes do programa) pode ser reutilizado em outras aplicações.
	<i>Interoperabilidade</i>	O esforço exigido para se acoplar um sistema a outro.

É importante realçar que dificilmente conseguimos obter medidas diretas dos fatores de qualidade anteriormente relacionados. Na realidade, McCall propõe que um conjunto de métricas seja definido e usado para desenvolver expressões para cada um desses fatores, de acordo com a seguinte relação:

$$F_q = c_1 \times m_1 + c_2 \times m_2 + \dots + c_n \times m_n$$

onde F_q é um fator de qualidade de software,

c_i são coeficientes de regressão e,

m_i são métricas que afetam o fator de qualidade.

Em McCall [McCall 77], podem ser encontradas as métricas m_i usadas para cada um dos fatores de qualidade anteriormente relacionados.

2.4.2 – A Norma NBR 13596

A Norma NBR 13596¹ (Avaliação de Produto de Software – Características de Qualidade e Diretrizes para seu Uso) [NBR 13596] define seis características (subdivididas em subcaracterísticas), que descrevem, com um mínimo de sobreposição, a qualidade de software. Essas características fornecem uma base para posterior refinamento e descrição de qualidade de software.

Esta Norma é dirigida àqueles envolvidos com aquisição, desenvolvimento, uso, suporte, manutenção ou auditoria de software.

2.4.2.1 – As Características de Qualidade da NBR 13596

Segundo a NBR 13596, a qualidade de um produto de software pode ser avaliada pelas características descritas a seguir.

- **Funcionalidade:** refere-se ao conjunto de atributos que evidenciam a existência de um conjunto de funções que satisfaz necessidades estabelecidas ou implícitas e suas propriedades especificadas. Suas subcaracterísticas são: Adequação, Acurácia, Interoperabilidade, Conformidade e Segurança de acesso (Tabela 2.2).

¹ Internacionalmente conhecida como ISO/IEC 9126 [ISO/IEC 9126].

Tabela 2.2 – Subcaracterísticas da Funcionalidade [NBR 13596]

Funcionalidade	
Subcaracterística	Descrição
Adequação	Atributos do software que evidenciam a presença de um conjunto de funções e sua apropriação para as tarefas especificadas.
Acurácia	Atributos do software que evidenciam a geração de resultados ou efeitos corretos ou conforme acordados.
Interoperabilidade	Atributos do software que evidenciam sua capacidade de interagir com sistemas especificados.
Conformidade	Atributos do software que fazem com que ele esteja de acordo com as normas, convenções ou regulamentações previstas em leis e descrições similares, relacionadas à aplicação.
Segurança de acesso	Atributos do software que evidenciam sua capacidade de evitar o acesso não autorizado, acidental ou deliberado, a programas e dados.

- Confiabilidade: refere-se ao conjunto de atributos que evidenciam a capacidade do software manter seu nível de desempenho, sob condições estabelecidas, por um determinado período de tempo. Suas subcaracterísticas são: Maturidade, Tolerância a falhas e Recuperabilidade (Tabela 2.3).

Tabela 2.3 – Subcaracterísticas da Confiabilidade [NBR 13596]

Confiabilidade	
Subcaracterística	Descrição
Maturidade	Atributos do software que evidenciam a frequência de falhas por defeitos no software.
Tolerância a falhas	Atributos do software que evidenciam sua capacidade em manter um nível de desempenho especificado nos casos de falhas no software ou violação nas interfaces especificadas.
Recuperabilidade	Atributos do software que evidenciam sua capacidade de restabelecer seu nível de desempenho e recuperar os dados diretamente afetados, em caso de falha, e o tempo e esforço necessários para tal.

- Usabilidade: refere-se ao conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o software, bem como o julgamento individual desse uso, por um conjunto explícito ou implícito de usuários. Suas subcaracterísticas são: Inteligibilidade, Apreensibilidade e Operacionalidade (Tabela 2.4).

Tabela 2.4 – Subcaracterísticas da Usabilidade [NBR 13596]

Usabilidade	
Subcaracterística	Descrição
Inteligibilidade	Atributos do software que evidenciam o esforço do usuário para reconhecer o conceito lógico e sua aplicabilidade.
Apreensibilidade	Atributos do software que evidenciam o esforço do usuário para aprender sua aplicação (por exemplo: controle de operação, entradas, saídas).
Operacionalidade	Atributos do software que evidenciam o esforço do usuário para sua operação e controle da sua operação.

- **Eficiência:** refere-se ao conjunto de atributos que evidenciam o relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados, sob condições estabelecidas. Suas subcaracterísticas são: Comportamento em relação ao tempo e Comportamento em relação aos recursos (Tabela 2.5).

Tabela 2.5 – Subcaracterísticas da Eficiência [NBR 13596]

Eficiência	
Subcaracterística	Descrição
Comportamento em relação ao tempo	Atributos do software que evidenciam seu tempo de resposta, tempo de processamento e velocidade na execução de suas funções.
Comportamento em relação aos recursos	Atributos do software que evidenciam a quantidade de recursos usados e a duração de seu uso na execução de suas funções.

- **Manutenibilidade:** refere-se ao conjunto de atributos que evidenciam o esforço necessário para realizar modificações específicas no software. Suas subcaracterísticas são: Analisabilidade, Modificabilidade, Estabilidade e Testabilidade (Tabela 2.6).

Tabela 2.6 – Subcaracterísticas da Manutenibilidade [NBR 13596]

Manutenibilidade	
Subcaracterística	Descrição
Analisabilidade	Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas.
Modificabilidade	Atributos do software que evidenciam o esforço necessário para modificá-lo, remover seus defeitos ou adaptá-lo a mudanças ambientais.
Estabilidade	Atributos do software que evidenciam o risco de efeitos inesperados ocasionados por modificações.
Testabilidade	Atributos do software que evidenciam o esforço necessário para validar o software modificado.

- **Portabilidade:** refere-se ao conjunto de atributos que evidenciam a capacidade do produto de software ser transferido de um ambiente para outro. O ambiente pode estar referenciando um ambiente organizacional, um ambiente de hardware ou um ambiente de software. Suas subcaracterísticas são: Adaptabilidade, Capacidade para ser instalado, Conformidade e Capacidade para substituir (Tabela 2.7).

Tabela 2.7 – Subcaracterísticas da Portabilidade [NBR 13596]

Portabilidade	
Subcaracterística	Descrição
Adaptabilidade	Atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado.
Capacidade para ser instalado	Atributos do software que evidenciam o esforço necessário para sua instalação num ambiente especificado.
Conformidade	Atributos do software que o tornam consonante com padrões ou convenções relacionados à portabilidade.
Capacidade para substituir	Atributos do software que evidenciam sua capacidade e esforço necessário para substituir um outro software, no ambiente estabelecido para esse outro software.

Para utilizar essas características com propósitos de definição e avaliação, é necessário estabelecer métricas, níveis de avaliação e critérios específicos para a organização e/ou aplicação.

A importância de cada característica de qualidade varia em função da classe da aplicação e do ponto de vista considerado (usuário, desenvolvedor ou gerente).

2.4.3 – Norma NBR ISO/IEC 12119

A Norma NBR ISO/IEC 12119¹ (Pacotes de Software – Teste e Requisitos de Qualidade) [NBR ISO/IEC 12119] é aplicável à avaliação de pacotes de software na forma em que são oferecidos e liberados para uso no mercado. É importante salientar que não é objetivo desta norma tratar o processo de produção do software, suas atividades e produtos intermediários ou o sistema de qualidade do produtor. Entende-se por pacote de software o "conjunto completo e documentado de programas fornecidos a diversos usuários para uma aplicação ou função genérica" (exemplos: processadores de texto, planilhas eletrônicas, bancos de dados, software gráficos, programas para funções técnicas ou científicas e programas utilitários).

Os potenciais usuários desta norma são fornecedores, entidades certificadoras, laboratórios de testes, compradores e usuários que podem se beneficiar com produtos melhor especificados. A Figura 2.2 mostra a estrutura básica da norma.

Figura 2.2 – Estrutura da Norma NBR ISO/IEC 12119 [NBR ISO/IEC 12119]

¹ Internacionalmente conhecida como ISO/IEC 12119 [ISO/IEC 12119].

2.4.3.1 – Requisitos de Qualidade

Um pacote de software deve possuir a Documentação do Pacote, que é composta pela *Descrição de Produto*, *Documentação de Usuário* e *Programa e Dados*. A seguir são descritos os requisitos de qualidade de cada um desses componentes.

Descrição de Produto

É um documento que expõe as principais propriedades de um pacote de software, com os seguintes objetivos:

- Auxiliar o usuário ou os potenciais compradores deste produto, na avaliação da adequação do produto às suas reais necessidades;
- Servir como base para testes.

Este documento deve estar disponível ao usuário, independentemente da aquisição do produto, através de um catálogo, de um disquete de apresentação ou qualquer outro meio disponível que alcance esse objetivo. A descrição deve ser clara, compreensível e harmônica com outros documentos associados. A norma propõe aspectos práticos e diretos, indicando "o quê" deve conter esta descrição.

Na Tabela 2.8, está resumida uma parte destas indicações, as quais podem ser *mandatárias* (termo "deve") ou *recomendáveis* (termo "pode"). É importante observar que o uso de um requisito como recomendável está diretamente relacionado com o tipo do produto, ou seja, para alguns tipos de produtos esses requisitos podem ser mandatórios (por exemplo, um sistema de reserva de passagens aéreas tem como um dos requisitos mandatórios a Eficiência) [NBR ISO/IEC 12119].

Tabela 2.8 – Requisitos de Qualidade para a Descrição de Produto [NBR ISO/IEC 12119]

Descrição de Produto	
Item	Requisitos
Requisitos Gerais sobre o Conteúdo da Descrição de Produto	O conteúdo da descrição deve ser inteligível, completo e possuir boa organização e apresentação, auxiliando os compradores em potencial na avaliação da adequação do produto às suas necessidades, antes de adquiri-lo.
Identificações e Indicações	Deve apresentar o nome do produto, sua versão, os requisitos de hardware e software, as principais atividades realizadas e os componentes entregues com o pacote.
Declaração sobre Funcionalidade	Deve apresentar uma visão geral das funções disponíveis, os valores limites se existirem e os dispositivos de segurança de acesso ao produto quando necessários.
Declaração sobre Confiabilidade	Deve apresentar as informações sobre os procedimentos para preservação de dados.
Declaração sobre Usabilidade	Deve apresentar o tipo de interface com o usuário, se é necessário algum conhecimento técnico específico para o seu uso e se o produto pode ser adaptado às necessidades do usuário.

Descrição de Produto (Continuação)	
Item	Requisitos
Declaração sobre Eficiência	Pode incluir informações a respeito do tempo de resposta e taxa de <i>throughput</i> .
Declaração sobre Manutenibilidade	Pode conter declarações sobre a manutenibilidade do produto.
Declaração sobre Portabilidade	Pode conter declarações sobre a portabilidade do produto.

Documentação de Usuário

É o conjunto completo de documentos, disponíveis na forma impressa ou não, que é fornecido para utilização de um produto, sendo também uma parte do produto. Ela deve incluir todos os dados necessários para a instalação (se necessário), para o uso da aplicação e para a manutenção do produto de software.

Os principais requisitos da *Documentação de Usuário* estão descritos na Tabela 2.9.

Tabela 2.9 – Requisitos de Qualidade para a Documentação de Usuário [NBR ISO/IEC 12119]

Documentação de Usuário	
Item	Requisitos
Compleitude	Deve conter todas as informações necessárias para o uso do produto, tal como estabelecer todas as funções do pacote, procedimentos de instalação e os valores limite.
Correção	A informação apresentada deve estar correta e sem ambigüidade.
Consistência	Deve haver plena coerência entre a documentação e a descrição de produto. Cada termo deve ter um único significado.
Inteligibilidade	A documentação deve ser compreensível pela classe de usuários que desenvolve atividades com o produto, utilizando termos apropriados, exibições gráficas e explicações detalhadas.
Apresentação e Organização	Deve ser apresentada através de uma forma que facilite uma visão geral, através de índices e tabelas de conteúdo. Se o documento não está na forma impressa, deve haver indicação de como efetuar a impressão.

Programas e Dados

Os requisitos de qualidade para *Programas e Dados* utilizam as mesmas definições das características de qualidade da Norma NBR 13596. As características de Funcionalidade, Confiabilidade e Usabilidade são destacadas e devem ser verificadas através do uso do produto. Não há requisitos específicos para os aspectos de Eficiência, Manutenibilidade e Portabilidade. Qualquer requisito declarado na documentação do pacote, referente às características citadas, deve estar em conformidade. Os principais requisitos para *Programas e Dados* estão descritos na Tabela 2.10.

Tabela 2.10 – Requisitos de Qualidade para Programas e Dados [NBR ISO/IEC 12119]

Programas e Dados	
Item	Requisitos
Funcionalidade	Devem ser verificados os procedimentos para instalação do produto; a presença de todas as funções mencionadas; a execução correta destas funções; a ausência de contradições entre a descrição do produto e a documentação do usuário.
Confiabilidade	O usuário deve manter o controle do produto, sem corromper ou perder dados, mesmo que a capacidade declarada seja explorada até os limites ou fora deles, se uma entrada incorreta é efetuada, ou ainda se instruções explícitas na documentação são violadas.
Usabilidade	A comunicação entre o programa e o usuário deve ser de fácil entendimento, através das entradas de dados, mensagens e apresentação dos resultados, utilizando um vocabulário apropriado, representações gráficas e funções de auxílio (<i>help</i>), entre outras; o programa também deve proporcionar uma apresentação e organização que facilite uma visão geral das informações, além de procedimentos operacionais que o auxiliem, por exemplo, a reversão de uma função executada e o uso de recursos de hipertexto em funções de auxílio, entre outras.

2.4.3.2 – Instruções para Teste

Este item recomenda como um produto deve ser testado em relação aos requisitos de qualidade. Inclui tanto o teste das propriedades necessárias a todos os produtos de mesmo tipo, quanto o teste das propriedades especificadas na descrição do produto. Na Tabela 2.11 são mostradas essas recomendações.

Tabela 2.11 – Instruções para Teste [NBR ISO/IEC 12119]

Instruções para Teste		
Fases	Componentes	Recomendações
Pré-Requisitos de Teste	<i>Presença de Itens do Produto</i>	Devem estar presentes, para a execução do teste, todos os componentes a serem entregues e os documentos de requisitos identificados na Descrição de Produto.
	<i>Presença de Componentes do Sistema</i>	Deve estar disponível todo o ambiente de hardware e software identificados na Descrição de Produto.
	<i>Treinamento</i>	Se o treinamento for mencionado na Descrição de Produto, o responsável pelo teste deve ter acesso ao material e ao programa de treinamento.
Atividades de Teste	<i>Descrição de Produto</i>	Todo requisito especificado nesta Descrição deve ser testado.
	<i>Documentação de Usuário</i>	Todo requisito especificado nesta Documentação deve ser testado.
	<i>Programas e Dados</i>	Todo requisito especificado para os Programas e Dados deve ser testado.
Registros de Teste		Os registros devem conter informações suficientes para permitir a repetição do teste, através de um Plano de Teste com os casos de teste, os resultados associados e a identificação das pessoas envolvidas.

Instruções para Teste (Continuação)		
Fases	Componentes	Recomendações
Relatório de Teste		Deve conter um resumo com os objetivos e resultados dos testes efetuados com a seguinte estrutura: identificação do produto; sistemas computacionais utilizados; documentos usados; resultados dos testes da Atividade de Teste; uma lista das não conformidades e a data de encerramento do teste.
Teste de Acompanhamento		Quando um produto é testado novamente (considerando o teste anterior), todas as partes modificadas e as partes inalteradas, mas influenciáveis pelas modificações, devem ser testadas como se fosse um produto novo.

2.4.4 – Norma ISO/IEC 14598-5

A Norma internacional ISO/IEC 14598-5 (Avaliação de Produto de Software – Parte 5: Processo para Avaliadores) [ISO/IEC 14598-5] fornece requisitos e recomendações para implementação prática da avaliação de produto de software. O processo de avaliação proposto pode ser usado para avaliar produtos já existentes ou produtos intermediários (em desenvolvimento). Pode ser utilizada por avaliadores de laboratório, fornecedores de software, compradores de software, usuários e entidades certificadoras, cada qual com seu objetivo.

Em termos de características de qualidade, pode ser usada a NBR 13596. Entretanto, o mensuramento direto dessas características não é prático. Nesse contexto, a utilização da experiência do engenheiro de software prevalece para a avaliação, o que pode reduzir a objetividade da avaliação [Tsukumo 97].

As características esperadas do Processo de Avaliação são [ISO/IEC 14598-5]:

- **Repetível:** a avaliação repetida de um mesmo produto, com mesma especificação de avaliação, realizada pelo mesmo avaliador, deve produzir resultados que podem ser aceitos como idênticos;
- **Reprodutível:** a avaliação do mesmo produto, com mesma especificação de avaliação, realizada por um avaliador diferente, deve produzir resultados que podem ser aceitos como idênticos;
- **Imparcial:** a avaliação não deve ser influenciada frente a nenhum resultado particular;
- **Objetiva:** os resultados da avaliação devem ser factuais, ou seja, não influenciados pelos sentimentos ou opiniões do avaliador.

O Processo de Avaliação proposto pela Norma ISO/IEC 14598-5 inclui cinco atividades: análise de requisitos da avaliação, especificação da avaliação, projeto da avaliação, execução da avaliação e conclusão da avaliação, como pode ser visto na Figura 2.3.

Figura 2.3 – Processo de Avaliação [ISO/IEC 14598-5]

A seguir são descritas as atividades do Processo de Avaliação.

Análise de Requisitos da Avaliação

Nesta fase deve-se descrever os objetivos da avaliação. Vários pontos de vista podem ser considerados, dependendo dos diferentes usuários do produto, tais como comprador, fornecedor, desenvolvedor e operador.

Especificação da Avaliação

Nesta fase deve-se definir o escopo da avaliação e as medidas a serem executadas no produto submetido à avaliação nos seus vários componentes. O nível de detalhes na especificação da avaliação deveria ser tal que, na sua base, a avaliação seja repetível e reprodutível.

Projeto de Avaliação

Nesta fase deve-se documentar os procedimentos a serem usados pelo avaliador para executar as medidas especificadas na fase anterior. O avaliador deve produzir um plano de avaliação, que descreve os recursos necessários para executar a avaliação especificada, assim como a distribuição desses recursos nas várias ações a serem executadas.

Execução da Avaliação

Nesta fase deve-se obter resultados da execução de ações para medir e verificar o produto de software de acordo com os requisitos de avaliação, como especificado na especificação da avaliação e como planejado no plano de avaliação. Ao executar essas ações, tem-se o rascunho do relatório de avaliação e os registros da avaliação.

Conclusão da Avaliação

Nesta fase, deve-se revisar o relatório da avaliação e disponibilizar os dados resultantes da mesma.

2.5 – Modelos de Avaliação e Melhoria de Processo de Software

Vários esforços e trabalhos têm contribuído para a elaboração de modelos e normas para avaliação e melhoria de qualidade em processos de software. Dentre esses os mais significativos pela sua disseminação e importância de contribuição são o Capability Maturity Model (CMM), a Norma NBR ISO/IEC 12207, o relatório técnico ISO/IEC TR 15504 (futura Norma ISO/IEC 15504) e a Norma Internacional ISO 9000-3 os quais são resumidamente apresentados nesta seção.

2.5.1 – Capability Maturity Model (CMM)

Maturidade de processo é o grau em que esse processo é definido, gerenciado e controlado. A capacidade de atingir metas de custo, qualidade e cronograma são influenciados pela maturidade do processo de software da organização. Assim, em adição ao modelo de processo, há questões relacionadas a todo sistema de gerenciamento de qualidade da companhia, o qual é importante para o resultado do projeto de software.

O modelo *Capability Maturity Model* (CMM) [Paulk 93] propõe a avaliação da capacidade e maturidade de uma organização e indica diretrizes para a melhoria. Foi desenvolvido pelo *Software Engineering Institute* (SEI), atendendo a clientes como o *Department of Defense* (DoD) dos EUA. Apresenta uma estrutura básica de métodos confiáveis e coerentes para a avaliação de qualidade de processo de software. Além disso, apresenta um modelo que apóia a melhoria do processo de software de uma organização através da aplicação criteriosa de conceitos de gerenciamento de processo e de melhoria de qualidade do desenvolvimento e manutenção de software [Marciniak 94].

No modelo CMM, os processos de software são classificados em cinco níveis de maturidade que representam a situação em que se encontra a organização em relação aos seus processos de software. Cada nível de maturidade é um estágio bem definido de evolução de processo de software, composto por um conjunto de atividades denominado áreas-chave de processo (ACPs) que, quando satisfeitas, aumentam a capacitação do processo de software [Humphrey 89].

Os cinco níveis de maturidade do modelo CMM e suas respectivas ACPs (com exceção do nível 1, que não possui área-chave de processo), são apresentados na Tabela 2.12.

Tabela 2.12 – Níveis e Áreas-Chave de Processo do Modelo CMM

Níveis	Enfoque	Áreas-Chave de Processo	<p>Qualidade Produtividade</p> <p>Risco Retrabalho</p>
5 Otimizado	<i>Melhoramento Contínuo do Processo</i>	Prevenção de Defeitos Gerenciamento de Mudança de Tecnologia Gerenciamento de Mudança de Processo	
4 Gerenciado	<i>Gerenciamento Quantitativo</i>	Gerenciamento Quantitativo do Processo Gerenciamento de Qualidade de Software	
3 Definido	<i>Padronização do Processo</i>	Foco no Processo da Organização Definição do Processo da Organização Programa de Treinamento Gerenciamento de Software Integrado Engenharia de Produto de Software Coordenação Intergrupos Revisões	
2 Repetitivo	<i>Gerenciamento de Projetos Básicos</i>	Gerenciamento de Requisitos Planejamento de Projeto de Software Acompanhamento e Supervisão do Projeto de Software Gerenciamento de Subcontrato de Software Garantia de Qualidade de Software Gerenciamento de Configuração do Software	
1 Inicial	<i>Sucesso é conseguido através de esforço pessoal</i>		Risco Retrabalho

A seguir são apresentadas as principais características de cada um dos níveis de maturidade do modelo CMM.

2.5.1.1 – Nível 1: Inicial

No nível inicial de maturidade, a organização não fornece um ambiente estável para o desenvolvimento e manutenção de software. Compromissos são assumidos pelos gerentes, sem que haja uma análise conjunta com os demais responsáveis pelo desenvolvimento do software sobre a possibilidade destes serem cumpridos [Tsukumo 96]. Assim cronogramas e orçamentos são freqüentemente abandonados por não serem baseados em estimativas realistas. Quando metas irreais são impostas, há normalmente um comprometimento da qualidade do produto com a eliminação ou diminuição de etapas de desenvolvimento, como testes e revisões, para que o cronograma possa ser cumprido [Curtis 92]. O sucesso depende totalmente das pessoas envolvidas no projeto, suas habilidades técnicas, conhecimento, motivação e do bom senso dos gerentes.

O processo de software de uma organização do nível 1 é imprevisível, já que o mesmo é constantemente alterado no decorrer do projeto. A funcionalidade e a qualidade do produto também não podem ser previstas. Existem poucos processos estáveis e que possam ser controlados na organização, não havendo mecanismos que permitam aprender com a experiência de projetos anteriores [Fiorini 98].

2.5.1.2 – Nível 2: Repetitivo

No nível repetitivo, políticas de gerenciamento do projeto de software e procedimentos para a implementação destas políticas são estabelecidos [Curtis 92]. O planejamento e gerenciamento de novos projetos são baseados na experiência obtida com projetos similares no passado [Humphrey 89]. Um dos objetivos do nível 2 é a institucionalização de um processo efetivo de gerenciamento do projeto de software, o qual permite que a organização repita o sucesso obtido em projetos anteriores. Um processo efetivo pode ser caracterizado como praticado pela organização, documentado, treinado (ensinado), medido, controlado e possível de ser melhorado. As estimativas são realistas, pois são baseadas na experiência da organização e nos requisitos do projeto em questão. Há um acompanhamento dos custos do projeto, cronograma, funcionalidade do produto, qualidade, etc. Os problemas são identificados e corrigidos. Caso haja uma terceirização de etapas do ciclo de vida de desenvolvimento do software, é estabelecida uma forte relação com o fornecedor (terceiro) de maneira que as atividades desenvolvidas pelo terceiro possam ser acompanhadas.

As organizações do nível 2 podem ser definidas como disciplinadas, pois o planejamento e o acompanhamento do projeto são claramente estabelecidos e controlados, permitindo a repetição de resultados de sucesso [Paulk 93].

2.5.1.3 – Nível 3: Definido

No nível definido, o processo de software padrão da empresa é documentado, incluindo as atividades referentes à engenharia de software e de gerenciamento do projeto de software. Um processo definido permite que atividades de engenharia de software e gerenciamento sejam integradas a cada projeto, auxiliando tanto técnicos como gerentes a realizarem suas atividades de maneira mais eficiente. Um grupo para processos de engenharia de software é estabelecido para facilitar atividades de definição e melhoria de processos. Um programa de treinamento é montado para garantir que os gerentes e os técnicos adquiram as habilidades e conhecimentos necessários para exercerem suas atividades.

As organizações do nível 3 podem ser definidas como sendo padronizadas e consistentes, já que as atividades de engenharia de software e gerenciamento são estáveis e podem ser repetidas. Dentro da linha de produtos estabelecida, custo, cronograma, e funcionalidade estão sob controle e a qualidade do software é acompanhada. A capacidade da organização é baseada num

conjunto comum de atividades, regras e responsabilidades de acordo com um processo de software definido [Paulk 93].

2.5.1.4 – Nível 4: Gerenciado

No nível gerenciado, a organização estabelece metas quantitativas de qualidade para os produtos e processos de software. Produtividade e qualidade são medidas para as atividades do processo de software em todos os projetos da organização. Um banco de dados é usado para coletar e analisar os dados decorrentes dos processos de software dos projetos. Os processos de software são instrumentados com medidas bem consistentes e definidas. Estas medidas estabelecem uma fundamentação quantitativa para a avaliação dos processos e produtos de software dos projetos. Os projetos adquirem controle sobre seus produtos e processos, através da diminuição da variação do seu desempenho de acordo com limites quantitativos aceitáveis [Fiorini 98].

As variações no processo de software podem ser distinguidas de variações aleatórias. Os riscos são diminuídos e controlados, pois há a aplicação de métricas de controle de qualidade.

As organizações do nível 4 são caracterizadas como previsíveis, pois o processo de software é medido e opera dentro de limites mensuráveis. Quando estes limites são ultrapassados, ações são tomadas para corrigir o problema e retornar à situação de normalidade. Os produtos de software são previsivelmente de alta qualidade [Paulk 93].

2.5.1.5 – Nível 5: Otimizado

No nível otimizado, a organização como um todo é voltado para a melhoria contínua do processo. A organização analisa periodicamente seus processos com o objetivo de prevenir a ocorrência de defeitos. Dados de eficiência do processo de software são utilizados para realização de análises de custo-benefício de implementação de novas tecnologias e proposição de mudanças nos processos de software da organização. As inovações em engenharia de software são identificadas e transferidas para toda a organização.

As equipes de projeto de organizações do nível 5 analisam defeitos para determinar suas causas. Os processos de software são avaliados para prevenir os defeitos conhecidos decorrentes de outros projetos passados.

As organizações do nível 5 podem ser caracterizadas como sendo continuamente melhoradas. Estas melhorias ocorrem tanto por incrementos nos processos de software existentes e praticados na organização como pela inovação e reestruturação total desses processos e a utilização de novas tecnologias e métodos [Paulk 93].

O nível 5 não é o destino final, e sim uma base para construir a capacidade de se estar sempre em aperfeiçoamento.

O CMM tendo se consolidado como um dos principais modelos de qualidade para processos de software, foi integrado (na sua versão 2.0 Draft C) juntamente com outros modelos (Electronic Industries Alliance/Interim Standard (EIA/IS) 731 e Integrated Product Development Capability Maturity Model (IPD-CMM) v.0.98) para formar o CMMI (Capability Maturity Model Integrated) [CMMI] o qual apresenta melhorias sugeridas pelos seus usuários, permitindo gerenciar novos aspectos para além do desenvolvimento de software, segundo conceitos de "sistema", englobando software, hardware e infra-estrutura. O CMMI está sendo desenvolvido como um modelo consistente com a futura Norma ISO/IEC 15504.

2.5.2 – Norma NBR ISO/IEC 12207

A norma NBR ISO/IEC 12207 [NBR ISO/IEC 12207] – Processos do Ciclo de Vida do Software tem o objetivo de estabelecer uma estrutura comum de processos de software, que seja utilizada como referência na contratação de produtos e serviços de software, bem como descrever as melhores práticas de engenharia e gerenciamento de software.

A norma faz uso de uma terminologia bem definida e sua estrutura é composta de processos, atividades e tarefas, a serem aplicadas em operações que envolvam o software de alguma forma, seja na aquisição, fornecimento, desenvolvimento, operação ou manutenção.

A estruturada norma foi concebida de maneira que fosse flexível, modular e pudesse ser adaptável à necessidade do produto a ser gerado. Para isto, ela baseia-se em dois princípios básicos: modularidade e responsabilidade. Entende-se modularidade como uma característica existente na definição dos processos, que possuem o mínimo de acoplamento e o máximo de coesão. Em princípio, todos os processos possuem uma única função no ciclo de vida. Define-se responsabilidade pela designação de uma parte envolvida para executar um processo, facilitando a aplicação da norma em um projeto, onde várias pessoas podem estar legalmente envolvidas [NBR ISO/IEC 12207].

Os processos da norma (Figura 2.4) são caracterizados em três tipos: Fundamentais, de Apoio e Organizacionais. A Norma não especifica como executar estes processos, deixando esta tarefa para a organização.

Figura 2.4 – Processos de Ciclo de Vida do Software [Rocha 01]

Os Processos Fundamentais formam o ciclo de vida do software, sendo responsáveis pela geração dos produtos de software. São representados pelos processos [NBR ISO/IEC 12207]:

- **Processo de Aquisição:** define as atividades do adquirente, organização que adquire um sistema, produto de software ou serviço de software.
- **Processo de Fornecimento:** define as atividades do fornecedor, organização que provê o sistema, produto de software ou serviço de software ao adquirente.
- **Processo de Desenvolvimento:** define as atividades do desenvolvedor, organização que define e desenvolve o produto de software.
- **Processo de Operação:** define as atividades do operador, organização que provê serviço de operação de um sistema computacional, no seu ambiente de funcionamento, para seus usuários.
- **Processo de Manutenção:** define as atividades do mantenedor, organização que provê o serviço de manutenção do produto de software, isto é, gerenciando as modificações no produto de software, isto é, gerenciando as modificações no produto de software para mantê-lo atualizado e em perfeita operação, até sua descontinuação.

Os Processos de Apoio têm como objetivo auxiliar outros processos, visando principalmente a qualidade e o sucesso do projeto. São representados pelos processos [NBR ISO/IEC 12207]:

- **Processo de Documentação:** define as atividades para registro da informação produzida por um processo de ciclo de vida.

- Processo de Gerência de Configuração: define as atividades de gerência de configuração
- Processo de Garantia da Qualidade: define as atividades para garantir objetivamente que os produtos e processos de software estão em conformidade com seus requisitos especificados e aderem aos seus planos estabelecidos.
- Processo de Verificação: define as atividades (para o adquirente, o fornecedor, ou uma parte independente) para verificação dos produtos de software, em profundidade variável, dependendo do projeto de software.
- Processo de Validação: define as atividades (para o adquirente, o fornecedor ou uma parte independente) para validação dos produtos de software do projeto de software.
- Processo de Revisão Conjunta: define as atividades para avaliação da situação e produtos de uma atividade. Este processo pode ser empregado por qualquer uma das duas partes, onde uma delas (parte revisora) revisa a outra parte (parte revisada) em um fórum conjunto.
- Processo de Auditoria: define as atividades para determinar a conformidade com requisitos, planos e contrato. Este processo pode ser empregado por qualquer uma das duas partes, onde uma delas (parte auditora) audita os produtos de software ou atividades da outra parte (parte auditada).
- Processo de Resolução de Problema: define um processo para análise e remoção dos problemas (incluindo não-conformidades), independente da sua natureza ou origem, que forem descobertos durante a execução dos processos de desenvolvimento, de operação, de manutenção ou de outros processos.

Os Processos Organizacionais têm como objetivo garantir e melhorar os processos dentro da organização. São representados pelos processos [NBR ISO/IEC 12207]:

- Processo de Gerência: define as atividades básicas da gerência, incluindo gerência de projeto, durante um processo de ciclo de vida.
- Processo de Infra-Estrutura: define as atividades básicas para o estabelecimento da estrutura de apoio de um processo de ciclo de vida.
- Processo de Melhoria: define as atividades básicas que uma organização (isto é, adquirente, fornecedor, desenvolvedor, operador, mantenedor, ou o gerente de outro processo) executa para estabelecer, medir, controlar e melhorar seu processo de ciclo de vida.
- Processo de Treinamento: define as atividades para prover pessoal adequadamente treinado.

Processo de Adaptação

O processo de adaptação define as atividades necessárias para adaptar a norma para sua aplicação na organização ou em projetos. A adaptação deve ser executada com base em alguns fatores que diferenciam uma organização ou projeto de outros, dentre os quais a estratégia de aquisição, modelos de ciclo de vida de projeto, características de sistemas e software e cultura organizacional. A existência desse processo permite que a norma seja adaptável a qualquer projeto, organização, modelo de ciclo de vida, cultura e técnica de desenvolvimento [Rocha 01].

Está prevista uma nova versão da norma NBR ISO/IEC 12207 para o início de 2002 que visa a retratar as melhores práticas adotadas pela comunidade de software.

2.5.3 – ISO/IEC TR 15504

Em junho de 1991, o grupo *International Organization for Standardization (ISO)* se reuniu para estudar um padrão para avaliação do processo de software. Firmou-se então um compromisso internacional de que o desenvolvimento do projeto seria coordenado por 4 centros técnicos de desenvolvimento: Inglaterra, Austrália, Canadá e EUA [Tsukumo 97].

Assim, em 1993 foi criado o *projeto SPICE (Software Process Improvement and Capability dEtermination)*, com o objetivo de gerar normas para avaliação de processos de software, visando a melhoria contínua do processo e a determinação da sua capacidade. O produto deste projeto é atualmente o relatório técnico ISO/IEC TR 15504 (futura Norma ISO/IEC 15504). O ISO/IEC TR 15504 foi projetado para satisfazer as necessidades dos clientes, dos fornecedores e dos avaliadores, e os requisitos individuais de cada um. Os benefícios incluem [ISO/IEC TR 15504]:

- Para os compradores: determinar a capacidade atual e potencial de um processo de software do fornecedor.
- Para fornecedores: determinar a capacidade atual e potencial de seus processos de software, efetuar a melhoria ou uma estrutura que define um roteiro para a melhoria do processo de software.
- Para avaliadores: uma estrutura que define todos os aspectos para conduzir avaliações.

A estrutura do ISO/IEC TR 15504 fornece meios para examinar e avaliar os processos de software usados pela organização. Dentro da visão do ISO/IEC TR 15504, os resultados da avaliação de processos de software podem ser usados tanto para a *determinação da capacidade dos processos*, quanto podem ser usados nas atividades de *melhoria de processo* (Figura 2.5) [ISO/IEC TR 15504].

Figura 2.5 – Avaliação de Processo de Software – ISO/IEC TR 15504 [ISO/IEC TR 15504]

No contexto da *determinação da capacidade do processo*, a avaliação fornece meios para identificar a capacidade de processos selecionados na unidade organizacional, tomando como base um processo idealizado. Essa análise comparativa serve tanto para identificar os riscos envolvidos em empreender um novo projeto, usando os processos selecionados, quanto de motivação para melhoria desses processos [Tsukumo 96].

No contexto de *melhoria de processo*, a avaliação fornece meios de caracterizar a prática atual aplicada na unidade organizacional, em termos da capacidade de processos selecionados, identificando mudanças que devem ser efetuadas para melhorá-los.

O ISO/IEC TR 15504 é composto por nove partes, e o contexto de cada uma delas é mostrado na Figura 2.6.

Figura 2.6 – Partes Componentes do ISO/IEC TR 15504 [ISO/IEC TR 15504]

A seguir são apresentadas cada umas das partes do modelo ISO/IEC TR 15504.

2.5.3.1 – Parte 1: Conceitos e Guia Introdutório

Esta parte fornece os conceitos de avaliação de processo de software e o uso desses conceitos no contexto de melhoria de processo e determinação da capacidade do processo. Descreve como as outras partes do modelo se integram e fornece diretrizes para sua seleção e uso. Explica os requisitos contidos no modelo e sua aplicabilidade para conduzir uma avaliação, para construção e seleção de ferramentas de suporte e para construção de um processo estendido. *Processos estendidos* ou são processos inteiramente novos, ou são processos que incluem práticas adicionais às práticas básicas, procurando adequar o processo às características específicas das organizações [ISO/IEC TR 15504a].

2.5.3.2 – Parte 2: Descrição de um Modelo Idealizado de Processo

A parte 2 do ISO/IEC TR 15504 define um modelo ideal de processo de software que forma a base para a avaliação. O modelo define, em alto nível, as atividades fundamentais que são essenciais para uma boa engenharia de software. O modelo descreve quais atividades são exigidas e não como devem ser implementadas. Esse modelo é aplicável a todas as empresas de software, não sendo necessário um tipo específico de estrutura organizacional, filosofia, modelo

de ciclo de vida do software, tecnologias ou metodologias de desenvolvimento [ISO/IEC TR 15504b].

2.5.3.3 – Parte 3: Classificação dos Processos

A parte 3 do ISO/IEC TR 15504 define um conjunto de requisitos mínimos para a condução da avaliação do processo, garantindo que as saídas desta avaliação sejam consistentes, possíveis de serem repetidas e que representem as instâncias do processo avaliado [ISO/IEC TR 15504c].

Esta parte define critérios para pontuar a capacidade de processos em relação ao processo idealizado. Além disso, nesta parte deve ser definido um formato de registro e saída das avaliações. Deve ser notado que o ISO/IEC TR 15504 não impõe qualquer formato específico para isto. Esta parte é importante para garantir que os resultados das avaliações sejam confiáveis, consistentes e reproduzíveis [ISO/IEC TR 15504c].

2.5.3.4 – Parte 4: Guia de Orientação para Condução da Avaliação

A parte 4 do ISO/IEC TR 15504 fornece um guia de orientação para conduzir uma avaliação. As diretrizes deste guia devem ser suficientemente genéricas para serem aplicáveis aos vários tipos de organização. Este guia fornece base para avaliar e interpretar os critérios definidos na parte 3 [ISO/IEC TR 15504d].

A avaliação de processo é usada para entender os processos atuais de uma organização. A avaliação se aplica a todos os processos relacionados (gerenciamento, desenvolvimento, manutenção, suporte) usados por uma organização.

2.5.3.5 – Parte 5: Diretrizes para Elaboração de um Instrumento de Avaliação

Um instrumento de avaliação é uma ferramenta, ou conjunto de ferramentas, usadas durante uma avaliação para auxiliar a estimativa da adequação ou da existência de práticas. Um instrumento de avaliação auxilia o avaliador fornecendo um conjunto consistente de indicadores que ajudam a julgar quão bem as práticas foram implementadas nos processos da organização. O instrumento de avaliação fornece um mecanismo para registrar as informações coletadas numa avaliação, sendo que estas podem ser recuperadas. A capacidade do instrumento de avaliação de armazenar e recuperar dados fornece a possibilidade de manter os resultados para posterior análise e melhoria [ISO/IEC TR 15504e].

O principal objetivo de um instrumento de avaliação é ajudar o avaliador a realizar uma avaliação de maneira consistente e reprodutível, reduzindo a subjetividade do avaliador e garantindo a validade, a usabilidade e a comparação entre os resultados da avaliação.

A parte 5 do relatório técnico ISO/IEC TR 15504 fornece diretrizes para construção de um instrumento de avaliação. Fornece um guia para seleção, construção e utilização dos instrumentos de avaliação.

Nesta parte não é fornecido um formato padrão para um instrumento de avaliação. O instrumento pode ser implementado como um questionário, um sistema especialista ou uma ferramenta manual ou on-line, já que os requisitos para um instrumento de avaliação são independentes de um projeto específico, estilo ou modo de uso [ISO/IEC TR 15504e].

2.5.3.6 – Parte 6: Qualificação e Treinamento de Avaliadores

Esta parte fornece orientação para a preparação e qualificação dos avaliadores para que estes realizem a avaliação do processo de software. Esta parte está relacionada com o treinamento, a experiência, a competência e ao aprendizado do avaliador, e também fornece mecanismos que podem ser usados para demonstrar competência e validar o aprendizado, o treinamento e a experiência dos avaliadores [ISO/IEC TR 15504f].

Este guia é direcionado primeiramente para os avaliadores, para os responsáveis pela seleção e formação dos avaliadores e pelos responsáveis por assegurar que um avaliador está apropriadamente qualificado para a tarefa. Também é útil para as organizações que desejam oferecer treinamentos apropriados de avaliação.

2.5.3.7 – Parte 7: Guia de Orientação para Melhoria de Processo

Esta parte fornece diretrizes para a utilização da avaliação de processo de software para o entendimento dos atuais processos de software da organização, e na utilização dos resultados da avaliação para formular ou priorizar os planos de melhoria. O guia inclui exemplos de aplicação de melhoria de processo numa variedade de situações. Os conceitos e princípios são apropriados para diferentes necessidades de negócios, domínios de aplicação e tamanho de organização, e eles podem ser usados por qualquer tipo de organização de software que deseje orientar suas atividades de melhoria [ISO/IEC TR 15504g].

Esta parte descreve como definir as entradas para avaliação do processo e como usar os resultados da avaliação para o propósito de melhoria de processo.

2.5.3.8 – Parte 8: Guia de Orientação para Determinação da Capacidade de Processo

Esta parte descreve como definir entradas para a avaliação do processo e como usar os resultados da avaliação para o propósito de determinar a capacidade do processo [ISO/IEC TR 15504h]. O guia especificamente direciona a determinação da capacidade do processo tanto para uso numa organização que deseja determinar os riscos associados pelo comprometimento com novo projeto, quanto pode ser usado por um comprador para avaliar fornecedores externos.

2.5.3.9 – Parte 9: Vocabulário

Esta parte contém um dicionário envolvendo todos os termos utilizados nas outras partes do ISO/IEC TR 15504. Os termos são primeiramente definidos em agrupamentos para facilitar o

entendimento. Os agrupamentos são feitos de maneira a manter termos comuns no mesmo grupo. Estes termos são apresentados em ordem alfabética para facilitar a referência [ISO/IEC TR 15504i].

2.5.4 – ISO 9000-3

Desde seu surgimento em 1987, a série de padrões ISO 9000 tem gerado bastante interesse nas indústrias e governos, em âmbito mundial. Este interesse é motivado principalmente pelo enfoque na qualidade, mas também, pela obrigatoriedade de certificação que muitas organizações e governos impuseram a seus fornecedores [Coallier 94].

A ISO 9000-3 [ISO 9000-3] é um guia para a aplicação da ISO 9001 [ISO 9001] no desenvolvimento, fornecimento e manutenção de software. A Norma ISO 9001 faz parte da série de normas ISO 9000, voltadas para a gestão e garantia da qualidade. Estas normas especificam os requisitos mínimos para que as empresas possam assegurar a qualidade de seus produtos e serviços, não definindo modelos ou impondo sistemas de qualidade a serem implementados nas organizações. As empresas definem seus próprios modelos de gestão de qualidade, dependendo do seu tipo de negócio e suas características [Tsukumo 97].

A ISO 9001 é aplicável a empresas em geral que atuam em projeto, desenvolvimento, produção, instalação e assistência técnica. Como esses documentos da Série ISO 9000 são genéricos, foi necessária a elaboração de um documento complementar onde fossem abordados alguns aspectos específicos do software.

Assim, em 1993 foi criada a *Norma ISO 9000-3 – "Diretrizes para Aplicação da ISO 9001 ao Desenvolvimento, Fornecimento e Manutenção de Software"*. Para cada item da ISO 9001 existe um correspondente na ISO 9000-3 que o detalha e o adapta ao software [Lamprecht 94]. As diretrizes propostas na ISO 9000-3 destinam-se a descrever os controles e métodos sugeridos para a produção de software que atendem aos requisitos do comprador, evitando-se não conformidades em todos os estágios, desde o desenvolvimento até a manutenção [ISO 9000-3]. A ISO 9000-3 é dividida em três partes, *estrutura*, *atividades do ciclo de vida* e *atividades de suporte* (Figura 2.7).

Figura 2.7 – Partes Componentes da ISO 9000-3 [Rosa 97]

2.5.4.1 – Estrutura

Descreve aspectos organizacionais, relacionados ao sistema de qualidade. São detalhadas as responsabilidades e ações relacionadas à qualidade que devem ser tomadas tanto pelo fornecedor quanto pelo comprador [ISO 9000-3]. Os pontos abordados são descritos na Tabela 2.13.

Tabela 2.13 – ISO 9000-3 (Estrutura) [ISO 9000-3]

ISO 9000-3 – Estrutura	
Est1 – Responsabilidade da Administração	
Definir e documentar sua política e objetivos para a qualidade, e assumir o compromisso com estes. O comprador deve cooperar com o fornecedor, a fim de prover todas as informações necessárias em tempo hábil e resolver itens pendentes.	
Est2 – Sistema da Qualidade do Fornecedor	
Estabelecer e manter documentado um sistema de qualidade. Este sistema deve ser um processo integrado por todo ciclo de vida. A prevenção de problemas deve ser enfatizada mais do que a correção.	
Est3 – Auditorias Internas do Sistema da Qualidade	
Implantar um sistema abrangente de auditorias internas de qualidade planejadas e documentadas, para verificar se as atividades de qualidade estão em conformidade com a forma planejada e para determinar a eficácia do sistema de qualidade. Os resultados das auditorias devem ser documentados para que se possa fazer as ações corretivas referentes às deficiências encontradas.	
Est4 – Ação Corretiva	
Estabelecer, documentar e manter procedimentos para investigar as causas de produto não conforme, detectar e eliminar causas potenciais, iniciar ações preventivas para tratar dos problemas, aplicar controles para assegurar a realização das ações corretivas e implementar e registrar alterações nos procedimentos resultantes de ações corretivas.	

2.5.4.2 – Atividades do Ciclo de Vida

Descreve as atividades de desenvolvimento de software. A Norma define que o desenvolvimento de software deve ser feito segundo um determinado modelo de ciclo de vida, e as atividades relacionadas à qualidade devem ser planejadas e implementadas de acordo com a natureza deste modelo utilizado [ISO 9000-3]. Independentemente do modelo de ciclo de vida estabelecido pela organização, a Norma define que as atividades do ciclo de vida devem ser agrupadas em nove categorias, descritas na Tabela 2.14.

Tabela 2.14 – ISO 9000-3 (Atividades do Ciclo de Vida) [ISO 9000-3]

ISO 9000-3 – Atividades do Ciclo de Vida	
Cic1 – Análise Crítica do Contrato	
Analisar cada contrato para assegurar que os objetivos, os requisitos e os riscos sejam definidos e documentados; que as informações reservadas sejam protegidas; que a terminologia seja aceita por ambas as partes; e que o comprador tenha capacitação para atender os requisitos contratuais. Deve-se considerar alguns itens como: os critérios de aceitação; o tratamento das alterações nos requisitos durante o desenvolvimento; recursos, ferramentas e itens de software fornecidos pelo comprador; normas e procedimentos a serem utilizados.	
Cic2 – Especificação dos Requisitos do Comprador	
Os requisitos devem incluir todos os aspectos necessários à satisfação das necessidades do comprador, como desempenho, segurança contra riscos, privacidade e confiabilidade. Tais requisitos devem ser estabelecidos de maneira precisa, o suficiente para permitir a validação durante a aceitação do produto, sendo que estes devem estar registrados na especificação. A especificação de requisitos também deve ter especificado todas as interfaces entre o produto de software e outros produtos de software ou de hardware.	
Cic3 – Planejamento do Desenvolvimento	
Definir um processo ou metodologia disciplinada, para transformar a especificação dos requisitos do comprador num produto de software. Isto envolve a identificação de:	
<ul style="list-style-type: none"> • Fases de desenvolvimento a serem realizadas; • Entradas requeridas para cada fase; • Saídas requeridas para cada fase; • Procedimentos de verificação a serem executados em cada fase; • Análise dos problemas potenciais associados às fases de desenvolvimento e ao atendimento dos requisitos especificados. 	
Cic4 – Planejamento da Qualidade	
Deve preparar um plano de qualidade como parte do planejamento de desenvolvimento. Este plano deve ser atualizado, ao longo da execução do desenvolvimento, deve conter atividades como análises críticas e ensaios, gestão de configuração e controle de alteração, e também controle de defeitos e ação corretiva.	
Cic5 – Projeto e Implementação	
Transformar as especificações dos requisitos do comprador num produto de software. Estas atividades devem ser realizadas de maneira disciplinada, a fim de produzir um produto de acordo com a especificação, e não depender das atividades de ensaio e de validação para assegurar a qualidade.	

ISO 9000-3 – Atividades do Ciclo de Vida (continuação)	
Cic6 – Ensaios e Validação	
Os resultados dos ensaios devem ser registrados, sendo que as áreas atingidas por quaisquer alterações devem ser identificadas e ensaiadas novamente. A adequação e pertinência dos ensaios devem ser avaliadas. Antes de liberar o produto, o fornecedor deve validar sua operação como um produto completo, quando possível, sob condições semelhantes às do ambiente de aplicação.	
Cic7 – Aceitação (pelo comprador)	
Julgar se o produto é aceitável ou não, conforme os critérios previamente acordados. O método para tratar dos problemas detectados deve ser documentado e acordado entre o fornecedor e o comprador.	
Cic8 – Cópia, Entrega e Instalação	
A cópia é uma etapa que deve ser conduzida antes da entrega. A cópia deve conter versão, manuais e guias do usuário, definição e acordo quanto aos direitos autorais e licenças. Prescrições devem ser feitas para verificar a correção e completeza das cópias dos produtos de software entregues. Na instalação as funções, responsabilidades e obrigações do fornecedor e do comprador devem ser claramente estabelecidas.	
Cic9 – Manutenção	
É estipulada no contrato, após a entrega e instalação inicial do produto de software. O fornecedor deve estabelecer e manter procedimentos para realizar atividades de manutenção e verificar se estas atividades atendem aos requisitos especificados para a manutenção. As atividades de manutenção são classificadas em: resolução de problemas, modificação de interface e expansão funcional ou aprimoramento de desempenho. Os itens nos quais a manutenção deve ser realizada e o período de tempo, durante o qual ela deve ser efetuada, devem ser especificados no contrato.	

2.5.4.3 – Atividades de Suporte

Descreve as atividades que apóiam as atividades do ciclo de vida de desenvolvimento. Estão organizadas em nove itens: gestão de configuração; controle de documentos; registros da qualidade; medição; regras, práticas e convenções; ferramentas e técnicas; aquisição; produto de software incluído e treinamento, descritos na Tabela 2.15.

Tabela 2.15 – ISO 9000-3 (Atividades de Suporte) [ISO 9000-3]

ISO 9000-3 – Atividades de Suporte	
Sup1 – Gestão de Configuração	
Fornecer um mecanismo para identificação, controle e acompanhamento das versões de cada item de software. O sistema de gestão de configuração deve: identificar as versões de cada item de software; identificar a situação de elaboração de produtos de software em desenvolvimento ou já entregues e instalados; controlar a atualização simultânea de um determinado item de software por mais de uma pessoa; identificar e seguir, do início até a liberação, todas as ações e alterações resultantes de uma solicitação de alteração.	
Sup2 – Controle de Documentos	
Estabelecer e manter procedimentos para controlar todos os documentos relacionados ao conteúdo da Norma ISO 9000-3. Os documentos abrangem: aprovação e emissão de procedimentos; procedimentos de alteração; entradas e saídas da fase de desenvolvimento; planos e resultados de verificação e validação; documentação para o comprador e o usuário e documentação de manutenção. Todos os documentos devem ser analisados criticamente e aprovados por pessoal autorizado, antes de serem emitidos.	

ISO 9000-3 – Atividades de Suporte (continuação)	
Sup3 – Registro da Qualidade	
Estabelecer e manter procedimentos para identificar, coletar, indexar, arquivar, armazenar, manter e dispor os registros da qualidade. Todos os registros da qualidade devem ser legíveis e identificáveis em relação ao produto envolvido. Os tempos de retenção dos registros da qualidade devem ser estabelecidos e registrados.	
Sup4 – Medição	
As medidas devem ser relatadas e utilizadas para gerenciar o processo de desenvolvimento e entrega, e devem ser pertinentes ao produto de software específico. O fornecedor deve dispor de indicadores quantitativos da qualidade do processo de desenvolvimento e entrega, os quais devem refletir: o quanto o processo de desenvolvimento está sendo efetuado em relação ao projeto e aos objetivos de qualidade conforme programado; com que eficácia o processo de desenvolvimento está reduzindo a probabilidade de que sejam introduzidos defeitos, ou que quaisquer defeitos introduzidos não sejam detectados.	
Sup5 – Regras, Práticas e Convenções	
Suprir regras, práticas e convenções de modo a tornar efetivo o sistema da qualidade especificado na Norma ISO 9000-3. O fornecedor deve analisar criticamente estas regras, práticas e convenções e revisá-las, quando requerido.	
Sup6 – Ferramentas e Técnicas	
Utilizar ferramentas, recursos e técnicas para tornar efetivas as diretrizes do sistema de qualidade. Estas ferramentas, recursos e técnicas podem ser eficazes para fins de gerenciamento, assim como para o desenvolvimento de produtos. O fornecedor deve aprimorar as ferramentas e técnicas, quando requerido.	
Sup7 – Aquisição	
Assegurar que um produto ou serviço adquirido está de acordo com os requisitos especificados. Os documentos de aquisição devem conter dados descrevendo claramente o produto ou serviço encomendado. O fornecedor deve analisar criticamente e aprovar os documentos de aquisição, quanto à adequação dos requisitos especificados antes da liberação.	
Sup8 – Produto de Software Incluído	
A inclusão ou uso de produtos de software fornecidos pelo comprador, ou por terceira parte, pode ser requerido ao fornecedor. O fornecedor deve estabelecer e manter procedimentos para validação, armazenamento, proteção e manutenção de tais produtos. Produtos fornecidos pelo comprador que sejam inadequados para o uso devem ser registrados e relatados ao comprador. A validação pelo fornecedor não isenta o comprador da responsabilidade de fornecer produtos aceitáveis.	
Sup9 – Treinamento	
Estabelecer e manter procedimentos para identificar as necessidades de treinamento, e providenciá-lo para todo o pessoal que executa atividades que influenciam a qualidade. Os assuntos a serem abordados devem ser determinados, considerando-se as ferramentas, técnicas, metodologias e recursos de computador específicos a serem usados no desenvolvimento e gerenciamento do produto de software. Também pode ser necessário incluir treinamento de habilidades e conhecimento do campo específico com o qual o software deve lidar.	

É importante salientar que a ISO 9001 somente indica o controle da não conformidade de um produto e recomenda ações corretivas e preventivas. A melhoria contínua do processo não é abordada por ela tal como é abordada explicitamente no modelo CMM [Coallier 94].

Atualmente a Norma ISO 9000-3 encontra-se em processo de revisão na ISO de forma a adequar-se à nova versão da ISO 9001:2000 [ISO 9001a].

2.6 – A Abordagem Goal/Question/Metric (GQM)

A qualidade representa um fator essencial no desenvolvimento de software, fato este que contribuiu para o surgimento de diversas abordagens de avaliação de qualidade. A principal característica que uma abordagem de avaliação de qualidade deve ter é sua capacidade de adaptação aos objetivos e particularidades de cada projeto, permitindo o aproveitamento de experiências, obtidas empiricamente, na melhoria de projetos futuros [Kirner 95][Kirner 97].

Esta seção apresenta a abordagem *Goal/Question/Metric* (GQM) [Basili 84][Basili 95], que foi originalmente proposta pelo professor Victor Basili, do *Software Engineering Laboratory* (SEL) – Universidade de *Maryland*, e que surgiu como um suporte na elaboração e implementação de programas de avaliação de atributos de qualidade de produtos e processos de software.

É um mecanismo simples e eficaz para estabelecer métricas, as quais podem ser planejadas, projetadas e implementadas a partir de um foco adequado e direto sobre as metas desejadas para o projeto de medição e que já foi utilizado em vários projetos de pesquisa de engenharia de software [Basili 86][Basili 88][Basili 90][Gresse 96][Rombach 87a][Rombach 89][Rombach 90][Rosa 97][Sanches 93][Shepperd 90]. Alguns exemplos de organizações onde foi aplicado são: NASA (National Aeronautics and Space Administration), Hewlett Packard, Motorola, Coopers & Lybrand.

2.6.1 – Visão Geral da Abordagem GQM

A abordagem GQM fornece um mecanismo que orienta a determinação dos objetivos de interesse da análise e o refinamento de cada objetivo em um conjunto de questões, com o propósito de quantificar os mesmos. Tais questões definem o conjunto específico de dados a serem coletados [Sanches 93].

O método GQM é apresentado na forma de uma estrutura hierárquica, que utiliza a técnica *top-down* (Figura 2.8), porque há muitas características observáveis no software, cuja forma de uso e interpretação só se tornam claras a partir da definição de modelos e objetivos apropriados para o contexto.

Figura 2.8 – Estrutura Hierárquica do Método GQM [Basili 95]

O método GQM consiste em 6 passos básicos, apresentados na Figura 2.9.

Figura 2.9 – Passos do Método GQM

2.6.2 – Estabelecer os Objetivos da Coleta de Dados

O primeiro passo focaliza o trabalho a ser realizado, ou seja, a definição das metas. A Meta é definida em termos do: *Objeto de Estudo*, *Propósito*, *Perspectiva*, *Ambiente*.

O *Objeto de Estudo* pode ser um processo, ou um produto, ou um recurso. Um processo está relacionado com as atividades de software normalmente associadas com o tempo, por exemplo, as atividades de especificação, projeto, teste e entrevista. Produtos são artefatos, documentos e produtos que são gerados durante o ciclo de vida do sistema, por exemplo: especificações, projetos, programas. Recursos são itens, usados pelos processos para produzir suas saídas, por exemplo: pessoal, hardware, software, espaço de escritório.

O *Propósito* pode ser: determinar, entender, melhorar, caracterizar, avaliar, prever, motivar ou controlar algum aspecto de qualidade do objeto de medida. Determina-se o porquê do projeto está sendo analisado.

A *Perspectiva* identifica a quem interessa os resultados da pesquisa; neste item fixa-se o ponto de vista da análise. Exemplo: ponto de vista do usuário, cliente, gerente, desenvolvedor, organização.

O *Ambiente* fornece o contexto para as interpretações dos resultados.

Como ilustrada na Figura 2.10, a meta segundo Basili [Basili 95], possui 3 coordenadas: Objetos, Propósitos e Pontos de Vista.

Figura 2.10 – Coordenadas da Meta do Método GQM [Basili 95]

Se as metas desejadas não forem fixadas ou não estiverem bem definidas, corre-se o risco de coletar dados que não estão relacionados com essas metas, dados sem significados para o objetivo da pesquisa, obtendo como resultado padrões incompletos ou não discerníveis.

2.6.3 – Desenvolver uma Lista de Questões de Interesse

Uma vez que as metas de estudo tenham sido estabelecidas, elas devem ser usadas para desenvolver uma lista de questões de interesse para ser respondidas pelo estudo. As questões de interesse definem os parâmetros e as categorias que permitem uma análise quantitativa dos dados.

As questões devem permitir uma caracterização do objeto (processo, produto ou recurso) de pesquisa e dos aspectos de qualidade do mesmo ponto de vista selecionado. As questões de interesse permitem uma definição mais nítida do objetivo da pesquisa, pois essas forçam os investigadores a considerar a análise de dados antes que eles sejam coletados [Rombach 87b].

A omissão de questões de interesse tem como consequência a falta de uma base quantitativa para satisfazer as metas dos estudos. Em efeito, se as questões de interesse não podem ser formuladas é porque as metas não foram bem definidas.

As metas resultarão na geração de várias questões de interesse diferentes e questões de interesse formam uma ponte entre determinar as metas subjetivamente do estudo e as medidas para serem usadas no estudo. Elas permitem aos investigadores determinarem as quantidades que precisam ser medidas e os aspectos das metas que podem ser medidos.

2.6.4 – Estabelecer as Métricas

Estabelecidas as questões de interesse o próximo passo é construir métodos de categorização. O passo 3 coloca como cada questão será avaliada, de modo que se possa saber quais os dados que devem ser coletados.

As métricas são definidas baseando-se nas abstrações, conceitos e propriedades associadas ao objeto de estudo.

Cada questão geralmente induz a um esquema de categorização que futuramente pode ser dividido em subcategorias. É importante ressaltar que cada medida necessita ser vista da perspectiva apropriada [Sanches 93]. As métricas podem ser objetivas e subjetivas.

As métricas objetivas são aquelas que dependem somente do objeto que esta sendo medido e não do ponto de vista da análise, por exemplo: número de versões do documento, horas do pessoal gasto em determinada tarefa, tamanho de um programa.

As métricas subjetivas são aquelas que dependem do objeto que está sendo medido e do ponto de vista do qual está sendo analisado, por exemplo: entendimento de um texto, nível de satisfação do usuário.

Se não for definido um esquema de categorização de dados antes da coleta, pode ser que mais tarde, eles não possam ser identificados, quando alguma categoria em particular for fixada.

2.6.5 – Projetar e Testar o Questionário de Coleta de Dados

Depois de estabelecidas as métricas, o questionário de dados é usado para proporcionar uma cópia permanente dos dados e reforçar a memória tanto do entrevistado como a do entrevistador.

A tarefa de projetar o questionário de coleta de dados foi uma das partes mais difíceis dos estudos conduzidos pelo *SEL*, principalmente porque os questionários representam um compromisso entre dois objetivos conflitantes.

Existe por parte do pesquisador o desejo de coletar um conjunto de dados completo e detalhado que conflitará com as necessidades de minimizar o tempo de esforço envolvido em fornecer esse conjunto de dados.

O primeiro objetivo conflitante permite a criação de questionários detalhados e extensos que requerem muito tempo para o seu preenchimento. O segundo objetivo conflitante referente ao tempo requer um questionário curto tipo teste de caixas.

A participação de fornecedores de dados (entrevistados) no processo de desenvolvimento do questionário é muito benéfica, porque as queixas relacionadas com o entendimento, por aqueles que devem usar o formulário são resolvidas mais cedo, antes da coleta de dados começar.

O questionário deve ser construído assim que os dados que eles contêm possam ser usados para responder às questões de interesse. Se necessário, as alterações no questionário de coleta de dados devem ser efetuadas antes que a coleta de dados comece. Um projeto de teste piloto é importante, pois várias iterações do projeto e períodos de testes são geralmente necessários, antes que um projeto satisfatório seja usado.

Se não for usado um questionário de coleta de dados é necessário confiar na memória do entrevistado e do entrevistador. Este enfoque pode permitir falta de completeza e dados inexatos.

2.6.6 – Coletar e Validar os Dados

Os dados são coletados através do preenchimento do questionário desenvolvido na etapa anterior.

A validação consiste em verificar os questionários quanto a não ocorrência de erros, consistência e completitude. Ela é baseada nas entrevistas com aquelas pessoas que fornecem os dados.

O resultado de coletar e validar os dados concorrentemente é que a exatidão do processo de coletar dados deve ser quantificada. A análise de dados do *SEL* mostra que sem a validação é possível a inexactidão dos dados na ordem de 50%.

2.6.7 – Analisar os Dados

O último passo é analisar e interpretar os dados coletados, no contexto das questões colocadas no passo 2 (seção 2.6.3) e com os quais os dados estão associados.

A análise dos dados varia significativamente de acordo com o propósito da medição. Um relacionamento estatístico deve ser identificado entre as métricas e o foco de qualidade, e esse relacionamento deve ser consistente com o objetivo especificado [Briand 96].

O sucesso de um programa de medição depende da sua aplicação prática. A maioria de falhas ocorridas neste tipo de iniciativa é a dificuldade de realização prática das métricas e seu efetivo uso, conforme se constata nas empresas que produzem sistemas de software. Isso decorre de um planejamento deficitário das métricas, como falta de aplicabilidade prática e a ineficácia nos resultados obtidos como indicadores de apoio aos processos técnicos e gerenciais.

2.7 – A Qualidade no Setor de Software Brasileiro

A preocupação mundial com a qualidade de software tem tido, recentemente, reflexos na produção brasileira de software. Uma vez que, para competir no mercado internacional, muitas vezes é necessário comprovar a qualidade de software, o setor de produção de software no Brasil tem passado a considerar esta importante questão. Uma prova concreta é a criação do Subcomitê Setorial de Qualidade e Produtividade de Software (SSQP/SW) no Programa Brasileiro da Qualidade e Produtividade (PBQP).

O grande objetivo do SSQP/SW-PBQP é atingir padrões internacionais de qualidade e produtividade e, para tal, várias estratégias e ações têm sido propostas no sentido de:

- Conscientizar as empresas de desenvolvimento de software da importância da qualidade;
- Oferecer infra-estrutura para que estas empresas busquem atingir padrões satisfatórios de qualidade.

Neste contexto, o SSQP/SW-PBQP tem realizado, desde 1993, pesquisas visando elaborar um diagnóstico periódico da qualidade e produtividade em software no Brasil. Já foram realizados quatro diagnósticos, em 1993, 1995, 1997 e 1999. A seguir são mostrados alguns dos resultados

obtidos no diagnóstico da qualidade e produtividade em software de 1999 [MCT 00], realizado com 446 empresas.

A partir da comercialização bruta anual proveniente de software, informada para o ano de 1998, as empresas foram classificadas por porte, de acordo com a faixa de valores a que pertenciam – microempresas (até R\$ 120 mil), pequenas (acima de R\$ 120 mil a R\$ 720 mil), médias (acima de R\$ 720 mil a R\$ 2,5 milhões) e grandes empresas (acima de R\$ 2,5 milhões). A predominância na amostra (63%) é de micro e pequenas empresas (Tabela 2.16).

Na atualidade, as economias mundiais passam por processos de globalização, com a liberalização dos mercados exigindo o crescimento e a modernização da indústria e da prestação de serviços, baseados não só na inovação e incorporação de novas tecnologias, nem no dinheiro, mas na capacidade gerencial das empresas que devem promover a competição de forma agressiva e em crescentes níveis de qualidade e produtividade [MCT 00].

Tabela 2.16 – Porte das Empresas, segundo Comercialização Bruta Anual Proveniente de Software [MCT 00]

Portes	Faixas de Comercialização	Nº	%	%
<i>Apropria receita específica</i>		343	77,2	100,0
Microempresas	Até R\$ 120 mil	96	21,6	29,4
Pequenas	Acima de R\$ 120 mil a R\$ 720 mil	112	25,2	34,2
Médias	Acima de R\$ 720 mil a R\$ 2,5 milhões	51	11,5	15,6
Grandes	Acima de R\$ 2,5 milhões	68	15,3	20,8
Não informaram valor		16	3,6	
<i>Não apropria receita específica</i>		38	8,6	
<i>Não comercializa software</i>		63	14,2	
Base		444	100	327

Planos estratégicos, planos de negócios ou planos de metas eram elaborados por 68% das empresas, sendo que destes, 46% com atualização sistemática e 54% sem periodicidade fixa para revisão, o que corresponde a 31% e 37% da amostra global, respectivamente (Tabela 2.17).

Tabela 2.17 – Elaboração de Planos Estratégicos, Planos de Negócios ou Planos de Metas [MCT 00]

Categorias	Nº de Empresas	%
Atualização sistemática	139	31,2
Revisão sem periodicidade fixa	164	36,9
Em implantação	84	18,9
Não elabora	58	13,0
Base	445	100

Dentre as 387 empresas que elaboravam ou estavam implantando seus planos, 69% incluíam metas ou diretrizes para a qualidade nesses planos, sendo que 62% dessas o faziam de forma sistemática e 38%, eventual, correspondendo, respectivamente, a 43% e 26%, em termos globais (Tabela 2.18).

Tabela 2.18 – Inclusão de Metas ou Diretrizes para Qualidade nos Planos [MCT 00]

Categorias	Nº de Empresas	%
Sistemática	165	42,6
Eventual	102	26,4
Pretende incluir	89	23,0
Não inclui	31	8,0
Base	387	100

Um grande número de empresários já reconhece que investimentos em qualidade produzem resultados extremamente positivos; não fazer nada é que custa caro, de tal modo que os prejuízos causados pela imagem de uma empresa associada à má qualidade podem ser incalculáveis [MCT 00].

Resultados sem diferenças significativas foram alcançados quanto ao conhecimento e uso de normas para qualidade de produtos (Tabela 2.19).

Tabela 2.19 – Conhecimento de Normas para Qualidade [MCT 00]

Categorias	NBR 13596		NBR ISO/IEC 12119	
	Nº	%	Nº	%
Conhece e usa	15	3,4	4	0,9
Conhece e começa a usar	27	6,1	23	5,2
Conhece, mas não usa	119	26,7	114	25,6
Não conhece	284	63,8	304	68,3
Base	445	100	445	100

Um percentual de 18% das empresas que conhecem as normas de qualidade de produto já teve produtos avaliados por terceiros ou adotavam procedimentos baseados em auto-avaliação (Tabela 2.20).

Tabela 2.20 – Avaliação de Produtos Baseados nas Normas de Qualidade de Produto de Software [MCT 00]

Categorias	Nº de empresas	%
Produtos avaliados por terceiros	12	8,6
Auto-avaliação	13	9,3
Em estudo ou implantação	46	32,9
Outras normas	18	12,9
Não adota	51	36,4
Base	140	100

O modelo CMM para classificação do sistema da qualidade de uma equipe ou empresa produtora de software, baseado na capacidade de seu processo de desenvolvimento, era conhecido por 47% das empresas, sendo usado por 21% dessas (Tabela 2.21).

Percentuais inferiores aos observados com relação ao modelo CMM de 30,7% para conhecimento e 3,4% para uso foram obtidos com relação ao relatório técnico ISO/IEC TR 15504 (futura Norma ISO/IEC 15504) (Tabela 2.21).

Tabela 2.21 – Conhecimento do modelo CMM e do relatório técnico ISO/IEC TR 15504 [MCT 00]

Categorias	CMM		ISO/IEC TR 15504	
	Nº	%	Nº	%
Conhece e usa sistematicamente	8	1,8	1	0,2
Conhece e começa a usar	36	8,1	14	3,2
Conhece, mas não usa	165	37,2	121	27,3
Não conhece	234	52,8	308	69,4
Base	443	100	444	100

No Brasil, há uma tendência da tecnologia de avaliação da qualidade dos produtos de software, baseada nas Normas NBR 13596, NBR ISO/IEC 12119 e ISO/IEC 14598, ser rapidamente disseminada entre as empresas de software. Isso se deve aos trabalhos do Centro Tecnológico para Informática (CTI), de Campinas, na escolha do Prêmio ASSESPRO (Associação das Empresas Brasileiras de Software e Serviços de Informática) de melhor software do ano e avaliação do software brasileiro que participa anualmente de feiras internacionais, como a COMDEX Fall, em Las Vegas, nos Estados Unidos, e a CeBIT, em Hannover, na Alemanha, sob coordenação do SOFTEX 2000 [Andrade 96].

Por isso, considera-se que a base para o reconhecimento da qualidade do software brasileiro e a chave para alcançar crescentes níveis de competitividade nas empresas brasileiras de software encontram-se na adoção [Weber 95][Weber 97]:

- De modelos de melhoria de processos de software, como o CMM;
- Da NBR 13596, NBR ISO/IEC 12119 e série ISO/IEC 14598 para avaliação dos produtos de software.

Através desses resultados notou-se que a maioria das empresas brasileiras de desenvolvimento de software encontra-se em um nível extremamente baixo no tocante à qualidade dos seus processos de desenvolvimento de sistemas e ao conhecimento de conceitos básicos de engenharia de software. Com a crescente disputa pelo mercado, essas empresas certamente precisarão adequar e evoluir seus produtos para atender as novas tecnologias e aos padrões atuais de qualidade.

2.8 – Considerações Finais

Neste capítulo foram introduzidos os conceitos de qualidade, qualidade de produto e de processo de software, modelos para avaliação de produtos de software, modelos para avaliação e melhoria de processo de software, a abordagem GQM e um panorama da qualidade no setor de software brasileiro. Observou-se que a maioria das empresas brasileiras de desenvolvimento de software ainda não tem uma cultura voltada a questão do uso de normas e modelos de qualidade de produto e processo de software.

Na questão de qualidade de software, a engenharia reversa tem um papel importante, auxiliando na atividade de manutenção e reengenharia de sistemas de software, desta forma permitindo que sistemas existentes possam ser melhor entendidos, documentados, reestruturados ou novamente desenvolvidos utilizando novas tecnologias e visando a atender padrões de qualidade. No próximo capítulo serão introduzidos os conceitos relevantes de engenharia reversa e um método de engenharia reversa orientado a objetos (FUSION-RE/I), que será utilizado como base para esse trabalho.

Capítulo 3 – Engenharia Reversa de Sistemas

3.1 – Considerações Iniciais

A fase de manutenção do ciclo de vida do software é a mais temida pois nela é necessário entender e alterar o código fonte de sistemas, na maioria das vezes desenvolvidos por outras pessoas, com diferentes estilos de programação. A troca desses sistemas é geralmente inviável tanto financeira quanto operacionalmente, uma vez que os usuários estão acostumados a operar o sistema e uma possível troca pode gerar transtornos, além de um certo tempo de treinamento e adaptação ao novo sistema. Assim, a necessidade por métodos, técnicas e ferramentas de engenharia de software que facilitem o entendimento de programa é essencial. Engenharia reversa visa analisar um software e representá-lo em um nível mais alto de abstração de tal forma que seja fácil de entender com o propósito, por exemplo, de manutenção de software, reengenharia, reuso ou documentação [Systa 00].

Neste capítulo são apresentados os conceitos relativos de engenharia reversa e o método de engenharia reversa orientado a objetos FUSION-RE/I, que será utilizado no escopo deste trabalho.

3.2 – Níveis de Abstração no Ciclo de Vida

O ciclo de vida de software refere-se às etapas da existência do software. Há diversos modelos de ciclo de vida, entre eles o modelo cascata, o evolutivo, o espiral [Pressman 00].

Independentemente do modelo de ciclo de vida adotado, o processo de desenvolvimento de software começa pela especificação dos requisitos, passa pelo projeto e termina na implementação do software (Figura 3.1)

No desenvolvimento de software, quando se parte dos requisitos, para a implementação, tem-se um aumento do nível de detalhes e, portanto uma diminuição na abstração.

Abstração é definida como a habilidade de se ignorar os aspectos de assuntos não relevantes para o propósito em questão, tornando possível uma concentração maior nos assuntos principais [Oxford 86]. A abstração relaciona-se com as atividades do ciclo de vida segundo os conceitos de níveis e graus de abstração:

- *Nível de abstração:* cada passo no processo de desenvolvimento de sistema é um refinamento do nível de abstração. Quanto mais próximo o projeto estiver da fase de especificação dos requisitos do sistema, maior será o nível de abstração do mesmo. Por outro lado quanto mais

próximo o projeto estiver do nível de implementação, menor será o nível de abstração do projeto. Geralmente no desenvolvimento de software, parte-se de informações em alto nível de abstração, e à medida que o projeto for evoluindo nas etapas do ciclo de vida, o nível de abstração vai diminuindo [Chikofsky 90];

- *Grau de abstração*: considerando-se o mesmo nível de abstração, as informações apresentadas de forma mais global possuem alto grau de abstração, enquanto as apresentadas de forma mais detalhada, baixo grau de abstração [Chikofsky 90].

O processo tradicional de engenharia reversa de software, caracterizado pelas atividades progressivas do ciclo de vida, que partem de um alto nível de abstração para um baixo nível de abstração, é conhecido como *engenharia progressiva*, enquanto o processo inverso, caracterizado pelas atividades retroativas do ciclo de vida, que partem de um baixo nível de abstração para um alto nível de abstração, é conhecido como *engenharia reversa* [Chikofsky 90].

Figura 3.1 – Níveis e Graus de Abstração no Ciclo de Vida [Costa 97]

3.3 – Definições de Engenharia Reversa

O termo *engenharia reversa* tem sua origem na análise de hardware, onde a prática de extrair projetos de produtos finalizados é comum. A engenharia reversa de hardware é regularmente aplicada para melhorar os produtos da própria empresa e também para analisar os produtos de concorrentes em situações militares ou de segurança nacional [Chikofsky 90].

O conceito de engenharia reversa de software é similar. Entretanto, o objetivo da engenharia reversa de hardware é, tradicionalmente, duplicar o sistema, enquanto o objetivo da engenharia reversa de software, freqüentemente, é obter um entendimento do sistema em um nível mais alto

de abstração e uma estrutura para auxiliar na manutenção, expansão ou substituição do sistema [Chikofsky 90].

Para Pressman [Pressman 00], a engenharia reversa de software é um processo de recuperação de projeto, consistindo em analisar um programa, num esforço para criar uma representação do mesmo, em um nível de abstração maior do que o código fonte.

Benedusi [Benedusi 92] define engenharia reversa de software como uma coleção de teorias, métodos e técnicas capazes de apoiar:

- O projeto e implementação de um processo para extrair e abstrair informações de sistemas existentes e produzir documentação consistente com o código;
- A adição de conhecimentos e experiências à documentação, os quais não podem ser automaticamente reconstruídos a partir do código.

Assim, de maneira geral, o objetivo da engenharia reversa é a produção de informações que possam aumentar o entendimento geral de sistemas de software.

Por sua vez, um software pode ser visualizado obtendo-se diferentes níveis de entendimento. Com base nos níveis de abstração, as visões obtidas do software são classificadas em quatro categorias: *visão em nível implementacional*, *visão em nível estrutural*, *visão em nível funcional* e *visão em nível de domínio* [Harandi 90].

- *Visão em nível implementacional*: abstrai características da linguagem de programação e especificamente da implementação;
- *Visão em nível estrutural*: abstrai detalhes da linguagem de programação para revelar sua estrutura a partir de diferentes perspectivas. O resultado é uma representação explícita das dependências entre os componentes do sistema;
- *Visão em nível funcional*: abstrai a função de um componente do sistema, isto é, o que o componente faz. Relaciona partes do programa às suas funções a fim de revelar as relações lógicas entre elas;
- *Visão em nível de domínio*: abstrai o contexto em que o sistema está operando, ou seja, o porquê do sistema ser desenvolvido.

As informações obtidas em um processo de engenharia reversa podem ser utilizadas em atividades como manutenção, teste, controle de qualidade de software e reengenharia. A reengenharia, também conhecida como renovação ou reaproveitamento, utiliza essas informações para alterar o software ou reconstituí-lo em uma nova forma, procedendo à implementação dessa reconstituição [Chikofsky 90]. É executada objetivando um projeto de

melhor qualidade, tenha ele a mesma funcionalidade ou implemente mudanças [Chikofsky 90] [Pressman 00].

3.4 – Categorias de Métodos de Engenharia Reversa

Para obter as diversas visões do software, usualmente é necessário acrescentar às informações contidas no código outras informações provenientes de conhecimento e de experiência humana. Baseado no nível de entendimento obtido do sistema e no escopo das informações fornecidas há uma categorização das técnicas de engenharia reversa: *redocumentação* e *recuperação de projeto* [Chikofsky 90].

Redocumentação: É a criação ou revisão de representações semanticamente equivalentes em um mesmo *nível de abstração* [Chikofsky 90]. O processo de *redocumentação* cria as representações a partir de informações obtidas apenas da análise do código fonte, embora a apresentação dessas informações possa se diversificar. As formas das representações são consideradas visões alternativas, cujo objetivo é melhorar a compreensibilidade global do sistema [Chikofsky 90].

A redocumentação não transcende a visão em nível estrutural e não atribui significados ao sistema analisado. Recuperações mais ambiciosas tais como função, os propósitos ou a essência do sistema exige um nível de entendimento maior e são definidas como *recuperação de projeto* [Chikofsky 90].

Recuperação de Projeto: Nesta categoria de engenharia reversa, o conhecimento do domínio das informações externas e as deduções são adicionadas às observações feitas sobre o sistema através do exame do mesmo, de modo a obter informações com nível mais alto de abstração [Chikofsky 90].

A recuperação de projeto recria abstrações do projeto a partir de uma combinação de código, documentação existente do projeto (se disponível), experiências pessoais, e conhecimentos gerais sobre o problema e o domínio de aplicação, ou seja, deve produzir todas as informações necessárias para a completa compreensão da funcionalidade do programa [Biggerstaff 89].

A Figura 3.2 apresentada as categorias de engenharia reversa, definidas pelas visões que elas obtém (implementacional, estrutural, funcional e de domínio) e o escopo das informações usadas (código fonte ou base de conhecimento).

Figura 3.2 – Visões das Categorias de Engenharia Reversa [Costa 97]

Nos últimos anos, há um crescente reconhecimento da importância de engenharia reversa tanto no campo acadêmico como nas empresas de desenvolvimento de software [Chikofsky 90][Choi 90][Benedusi 92][Sneed 95][Klösch 96]. Esse reconhecimento tem resultado na apresentação de diversas iniciativas abordando diferentes métodos, técnicas e ferramentas de engenharia reversa (e reengenharia), mas ainda poucos métodos descrevem suporte automático, planejamento e controle de sua aplicação (que é um “processo de desenvolvimento”) de maneira a permitir a avaliação de sua qualidade (processo e produtos desse processo).

Jacobson e Lindström [Jacobson 91] sugeriram reengenharia de sistemas antigos para a arquitetura orientada a objetos, tendo como objetivo mostrar como um método orientado a objetos pode ser usado para modelar sistemas legados não desenvolvidos com esse paradigma.

Já em Gall e outros [Gall 94], os autores afirmam que a transformação de programas procedimentais em programas orientados a objetos é um processo importante para aumentar o potencial de reuso dos programas procedimentais. Propõem um processo de transformação, conhecido como COREM (Capsule Oriented Reverse Engineering Method), que usa conhecimento do domínio da aplicação.

Gall e Klösch [Gall 95], relatam que identificar objetos em programas procedimentais e tornar suas dependências explícitas colaboram para o entendimento do projeto do sistema, evita degradação dos projetos originais durante a fase de manutenção e facilita o processo de reusabilidade. Além disso, essa identificação de objetos constitui base para uma completa rearquitetura do sistema, de um programa procedimental para um programa orientado a objetos.

Klösh discute em [Klösh 96] a importância da engenharia reversa para a realização de manutenção preventiva, corretiva e adaptativa nos sistemas legados das empresas; também existem outras aplicabilidades dessa abordagem como: diminuição dos custos de manutenção, melhorias na qualidade do sistema, vantagens competitivas ou facilidade no reuso de software. Cita a dificuldade que a equipe de engenharia reversa encontra na análise de grandes e complexos sistemas que foram criados sem técnicas de desenvolvimento, como: programação estruturada, orientação a objetos, etc.

Penteado, em [Penteado 96], propõe uma abordagem de engenharia reversa (FUSION/RE) para obter modelos de análise orientados a objetos a partir de sistemas desenvolvidos com orientação a procedimentos. Os modelos criados com a utilização do método FUSION/RE são baseados nos modelos da fase de análise do método Fusion [Coleman 96].

Tilley [Tilley 98] descreve um framework usado na engenharia reversa para auxiliar no entendimento de programas. O framework é baseado sobre um modelo descritivo que caracteriza importantes mecanismos de apoio baseado sobre uma hierarquia de atributos. Os atributos incluem suporte a modelo cognitivo, tarefas de engenharia reversa, atividades canônicas (que são características do processo de engenharia reversa), atributos de qualidade caracterizados pelo ambiente de engenharia reversa, e outras características.

Em Bennett [Bennett 98] é proposto um método para discussão de como abordagem de engenharia reversa podem ser avaliadas e comparadas.

Cimitile [Cimitile 99] apresenta um método para decomposição de sistemas legados em objetos. A identificação de objetos é centrada em torno de dados persistentes, como arquivos e tabelas em banco de dados, enquanto programas e rotinas são candidatos para implementação dos métodos dos objetos.

Em Systs [Systs 00] é descrito uma técnica de engenharia reversa para sistemas desenvolvidos na linguagem Java e implementa um protótipo de um ambiente para engenharia reversa denominado Shimba que está sendo construído para apoiar o entendimento de sistemas desenvolvidos em Java através da combinação de métricas e de programas de visualização.

A necessidade da tecnologia de engenharia reversa e reengenharia orientada a objetos tem sido reconhecida por muitas empresas européias. Essas empresas junto com algumas universidades desenvolveram um projeto de pesquisa chamado FAMOOS (Framework-based Approach for Mastering Object-Oriented Software Evolution) com o objetivo de investigar ferramentas e técnicas para lidar com sistemas legados orientados a objetos [FAMOOS 01].

A Figura 3.3 ilustra um processo de engenharia reversa descrito em Rosenberg [Rosenberg 00], o qual se inicia extraíndo os requisitos e informações do projeto a partir do código fonte e documentos existentes. Um documento de requisitos é produzido e uma abstração do projeto

(em alto nível) é extraída e expressa usando diagramas de fluxo de dados e diagramas de fluxo de controle. O projeto recuperado é então revisado para verificação de consistência e correitude.

Figura 3.3 – Um Processo de Engenharia Reversa [Rosenberg 00]

Muitas empresas têm incorporado a tecnologia orientada a objetos no seu processo de desenvolvimento de software com a perspectiva de aprimorar a qualidade dos produtos de software. Com a crescente aceitação da orientação a objetos, o número de pesquisas sobre esta abordagem aumentou.

Para aproveitar os benefícios que a abordagem orientada a objetos fornece, muitos estudos foram e estão sendo realizados no campo da engenharia reversa. Atualmente existem diversas abordagens de engenharia reversa orientada a objetos [Jacobson 91][Gall 94][Penteado 96][Costa 97][Cimitile 99][Systa 00]. A abordagem FUSION-RE/I [Costa 97] é uma delas, e gera documentação de análise orientada a objetos de sistema desenvolvido originalmente com orientação a procedimentos e será descrita na próxima seção.

3.5 – Método de Engenharia Reversa FUSION-RE/I

Para auxiliar a atividade de manutenção de software, foi desenvolvido no ICMC-USP por Costa [Costa 97] um método de engenharia reversa, denominado FUSION-RE/I (FUSION – Reverse Engineering/Interface) baseado nos conceitos e idéias do método FUSION-RE [Penteado 96] de engenharia reversa orientado a objeto. Esse método auxilia a atividade de manutenção de software, recuperando o modelo de análise de um sistema usando uma abordagem orientada a

objetos. Esse método tem sido aplicado com êxito em diversos sistemas e descritos em vários projetos de mestrado [Costa 97][Quinaia 98][Feltrim 99a][Jubileu 99]. Feltrim [Feltrim 99a] realizou um estudo da adaptação do método FUSION-RE/I para a notação UML (Unified Modeling Language) e publicou um relatório técnico contendo todos produtos obtidos da aplicação do método FUSION-RE/I a um sistema [Feltrim 99b]. Todos esses fatores foram levados em consideração na escolha do método de engenharia reversa a ser modelado segundo a abordagem SQUID.

O método FUSION-RE/I fornece mecanismos para abstrair visões funcionais e estruturais a partir de aspectos operacionais e de dados disponíveis via interface usuário-computador. Preocupa-se em investigar e propor passos seguros e definidos, como mostrados na Figura 3.4 e descritos no decorrer desta seção, para que um sistema desenvolvido sem a abordagem de orientação a objetos possa ser recuperado no nível de modelagem de análise, usando um método orientado a objetos.

Figura 3.4 – Síntese do Método de Engenharia Reversa FUSION-RE/I [Costa 97]

Esse método é constituído de duas etapas: na primeira, parte-se de considerações lógicas, obtidas por meio de aspectos operacionais e visões de dados disponíveis via interface, para a recuperação de visões funcionais do sistema; na Segunda etapa, parte-se de considerações físicas, obtidas por meio de código fonte, e das visões funcionais para a recuperação de visões estruturais do sistema.

A etapa de recuperação de visões funcionais é subdividida em dois passos: o primeiro envolve a reunião de todas as informações existentes sobre o sistema em estudo. Nesse passo deve-se [Costa 97]:

- Reunir toda a documentação envolvendo a informação textual existente sobre o sistema (manuais, livros, artigos, listagem do código fonte, etc.) e a informação técnica relacionada aos conceitos pertinentes (domínio do sistema, aplicação, linguagem de implementação do sistema, da interface, método de desenvolvimento, etc.);
- Identificar os usuários do sistema, pois informações importantes podem não estar documentadas e um modo de recuperá-las é a partir do conhecimento dos desenvolvedores e usuários. Esses conhecedores do sistema também são requeridos para auxiliar na validação dos modelos obtidos no processo de aplicação do método FUSION-RE/I;
- Analisar os dados a fim de identificar informações relacionadas aos requisitos do sistema, ao projeto arquitetural, de dados e procedimental, ao ambiente onde o sistema é executado, à linguagem de implementação do sistema, à organização dos arquivos no disco, etc.

Neste passo, uma abordagem de aquisição de conhecimento como descrito em Jubileu [Jubileu 99], pode ser implementada auxiliando na obtenção de informações pertinentes ao sistema objeto do processo de engenharia reversa.

O segundo passo da etapa recuperação de visões funcionais do método FUSION-RE/I envolve recuperar informações da fase de análise do sistema em estudo. Para representar essas informações são utilizados os modelos da fase de análise do método Fusion [Coleman 96] de desenvolvimento de software orientado a objeto. A fase de análise do Fusion gera dois modelos:

- *Modelo de objetos*, que descreve a estrutura do sistema;
- *Modelo de interface*, que descreve o comportamento do sistema.

O *modelo de interface* é composto de dois modelos que capturam diferentes aspectos do comportamento: o *modelo de ciclo de vida*, o qual caracteriza seqüências permitidas de operações e eventos do sistema, e o *modelo de operações*, o qual caracteriza o efeito de cada operação do sistema em termos de mudanças de estado e eventos gerados [Coleman 96]. Esse passo consiste em:

- Elaborar o *modelo de ciclo de vida* do sistema.

Para elaborar o modelo de ciclo de vida é necessário um bom conhecimento do domínio e da aplicação do sistema submetido à engenharia reversa, incluindo estudo da documentação existente e entrevistas com os usuários do sistema. A partir disso, e principalmente do uso do sistema, é possível definir as seqüências de operações permitidas e os eventos de entrada e de

saída que o sistema aceita [Costa 97]. O modelo de ciclo de vida especifica através de expressões regulares, as seqüências permitidas de interações (Tabela 3.1). O modelo de ciclo de vida possui a seguinte sintaxe geral:

ciclo de vida [Nome:] Expressão_regular
(Nome_Local = Expressão_Regular)*

Tabela 3.1 – Operadores das Expressões Regulares do Modelo de Ciclo de Vida [Coleman 96]

Alfabeto:	Eventos de entrada e eventos de saída (precedidos por um #)
Operadores:	Sejam x e y expressões do ciclo de vida, então: x.y denota que x é seguido de y x y denota que x ou y ocorre x* denota zero ou mais ocorrências de x x+ denota uma ou mais ocorrências de x [x] denota que x é opcional x y significa a intercalação arbitrária dos elementos de x e y
Substituição:	Uma expressão pode ser rotulada em uma substituição: Nome = expressão do ciclo de vida Obs: Nome pode ser usado em outras expressões, mas substituições não devem ser recursivas
Precedência de Operadores:	Em ordem decrescente a precedência é: [], *, +, ., , Obs: Parênteses podem ser usados para alterar a precedência

- Elaborar o *modelo de operações do sistema*

Para elaborar o modelo de operações do sistema, parte-se do modelo de ciclo de vida, obtido anteriormente. O modelo de ciclo de vida apresenta uma visão geral da funcionalidade das operações do sistema, as quais, para elaboração do modelo de operações, devem ser melhor estudadas, por meio do uso intensivo do sistema, de modo que possam ser especificadas detalhadamente [Costa 97]. A tabela abaixo apresenta a notação do modelo de operações para a descrição da operação.

Tabela 3.2 – Notação do Modelo de Operações [Coleman 96]

Operação	<i>Nome</i>	Identificador único da operação.
Descrição	<i>Texto</i>	Descrição informal e concisa da operação.
Lê:	<i>Itens</i>	Indica os valores que a operação pode ter acesso, sem modificar. A palavra <i>supplied</i> precedendo um item, em Lê , indica que o identificador é um parâmetro da operação.
Modifica:	<i>Itens</i>	Indica os valores que a operação pode ter acesso e alterar. A palavra <i>new</i> precedendo um item indica que a operação cria um novo objeto na base de dados do sistema.
Envia:	<i>Agentes e Eventos</i>	Compreende o nome de um agente e uma lista de eventos que podem ser enviados a ele pela operação.

Assume:	<i>Condição</i>	É um predicado que define a pré-condição.
Resultado:	<i>Estado</i>	É um predicado que define a pós-condição, relacionando o estado do sistema antes da operação ser invocada e o estado posterior à invocação.

- Desenvolver o *modelo de objetos*

Para se desenvolver o modelo de objetos do sistema deve-se primeiramente definir *assuntos* com os quais a funcionalidade do sistema se relaciona. O objetivo da definição de assuntos é identificar os diferentes *níveis de abstração* da funcionalidade do sistema e os possíveis agrupamentos de operações de um mesmo assunto. Seguindo a nomenclatura do método FUSION-RE [Penteado 96], esses assuntos são denominados *temas*. A Figura 3.5 mostra a notação do Modelo de Objetos segundo o método Fusion.

Figura 3.5 – Notação para o Modelo de Objetos segundo o Método Fusion [Coleman 96]

Quanto à etapa de recuperação de visões estruturais do método FUSION-RE/I os passos a serem desenvolvidos são subdivididos em [Costa 97]:

- 1) Elaborar o *quadro de procedimentos*;

2) Elaborar o *quadro das operações/procedimentos de implementação*.

O primeiro passo da etapa de recuperação das visões estruturais envolve entender a implementação do sistema, por meio da identificação dos procedimentos de implementação, da funcionalidade de cada procedimento identificado e a sequência de chamadas desses procedimentos [Costa 97]. A apresentação dos procedimentos de cada arquivo do sistema é realizada por meio de um conjunto de *quadros de chamadas* o qual vem acompanhado por um *índice geral dos procedimentos do sistema*. Esse passo, então, envolve:

- Elaborar o *quadro de chamadas de procedimentos*

Esse subpasso é semelhante ao primeiro passo do método FUSION-RE [Penteado 96], no qual a arquitetura do sistema é revitalizada [Costa 97]. Para cada arquivo do sistema deve ser elaborado um *quadro de chamadas dos procedimentos de implementação*. Nesse *quadro de chamadas* devem ser apresentados todos os procedimentos contidos no arquivo, fazendo-se referências aos procedimentos utilizados (CHAMA) e (CHAMADO POR);

Tabela 3.3 – Notação do Quadro de Chamadas de Procedimentos [Costa 97]

Diretório/Arquivo: Dir/Arq	
Descrição do arquivo Arq do diretório Dir	
Procedimento1 (dir/arq)	Descrição: Descrição do procedimento1(funcionalidade do procedimento) Chama: Procedimento utilizado1(dir/arq) Procedimento utilizado2(dir/arq) Chamado por: Procedimento utilizador3 (dir/arq)

- Elaborar o *quadro índice de procedimentos*

Para isso utilizam-se os quadros de chamadas de procedimentos de todos os arquivos. Esses quadros são reunidos, retirando-se deles, o cabeçalho e a coluna contendo a *Descrição* e os procedimentos *Chama* e *Chamado por* (Tabela 3.4).

Tabela 3.4 – Quadro Índice dos Procedimentos [Costa 97]

Sistema:		
Descrição do Sistema		
Procedimentos		Diretório/Arquivo
Procedimento1		Diretório1/arquivo1

O segundo passo da etapa *recuperar visões estruturais* do método FUSION-RE/I, envolve identificar, entre todos os procedimentos considerados no *quadro de chamadas*, aqueles que

implementam as operações da interface do sistema (identificadas na primeira etapa), e alocá-los, de acordo com sua funcionalidade, à interface ou a um dos temas do sistema (identificados na primeira etapa). Nesse passo, um quadro de operações/procedimentos de implementação deve ser elaborado (Tabela 3.5). Nesse quadro, as duas primeiras colunas são referentes às opções e operações da interface do sistema. A terceira coluna refere-se aos procedimentos que implementam as opções e as operações do sistema. As demais colunas são utilizadas para indicar a funcionalidade de cada procedimento, associando-os à interface ou a algum dos temas do sistema (identificados na etapa anterior) [Costa 97].

Tabela 3.5 – Quadro Operações/Procedimentos de Implementação [Costa 97]

Interface		Procedimentos	Funcionalidade			
Opções	Operações		Interface	Tema1	Tema2	...
Opção1	Operação 1.1	Proced. 1.1		X		
		Proced. 1.1.1	x			
		Proced. 1.1.2		X		
Opção2	Operação 2.1	Proced. 1.2		X		
		Proced. 1.2.1	x			
		Proced. 1.2.2			x	

Durante este trabalho foi efetuada a engenharia reversa parcial de um sistema desenvolvido em Clipper, contendo 12 módulos (*.PRG) totalizando 8545 linhas de código, 7 arquivos de cabeçalho (*.CH) contendo 551 linhas de código, 26 tabelas (*.DBF), 41 arquivos de índice (*.NTX) e 97 opções de menu. Essa atividade deu subsídios para um maior entendimento do processo de engenharia reversa (atividades e artefatos produzidos por esse processo) e dos métodos de engenharia reversa orientados a objetos FUSION-RE [Penteado 96] e FUSION-RE/I [Costa 97], além do próprio modelo de análise do método de desenvolvimento orientado a objetos Fusion [Coleman 96].

3.5.1 – Processo de Engenharia Reversa FUSION-RE/I Segundo a Notação UML

Como a UML (Unified Modeling Language) é hoje a notação padrão adotada pela maioria das empresas de desenvolvimento de software internacionais (com uma crescente aceitação pelas empresas nacionais) na modelagem de seus sistemas (principalmente aqueles desenvolvidos segundo o paradigma orientado a objetos), e o método FUSION-RE/I utiliza a notação estabelecida pelo método Fusion, Feltrim [Feltrim 99a] realizou um estudo para adaptar o método FUSION-RE/I à notação UML. Nesta subseção serão apresentados resumidamente os diagramas estabelecidos pela UML e resultados desse estudo de adaptação.

3.5.1.1 – A Linguagem de Modelagem Unificada (UML)

A UML é uma linguagem padrão para a visualização, a especificação, a construção e a documentação de artefatos de um sistema complexo de software [Booch 00]. Sendo uma linguagem, a UML tem uma sintaxe e uma semântica bem-definidas. A parte mais visível da sintaxe da UML é sua notação gráfica. Ela fornece vários tipos de diagramas que podem ser usados para visualizar e modelar o sistema de software a partir de diferentes perspectivas e/ou em diferentes níveis de abstração. UML apóia todos os estágios do ciclo de vida do processo de engenharia progressiva desde a especificação dos requisitos até a implementação e teste. Os mesmos tipos de diagramas usados na engenharia progressiva também estão sendo utilizados para propósitos de engenharia reversa [Systa 00].

A UML propõe nove diagramas e um conjunto de elementos que podem ser utilizados em quaisquer desses diagramas. Entre esses elementos encontram-se notas (símbolo para representar restrições e comentários anexados a um elemento ou a uma coleção de elementos), pacotes (mecanismo de propósito geral para a organização de elementos em grupos), estereótipos (permite a criação de novos tipos de blocos de construção que são derivados dos já existentes, mas específicos a determinados problemas), valores atribuídos (permite a criação de novas informações na especificação de um elemento), restrições (permite acrescentar novas regras ou modificar as já existentes), ou seja, elementos independentes da natureza do diagrama.

UML fornece diagramas que capturam informações sobre a estrutura estática do software, e diagramas que modelam o comportamento dinâmico do software.

Um diagrama é uma apresentação gráfica de um conjunto de elementos, em geral representado como um gráfico conectado de vértices (itens) e arcos (relacionamentos). São desenhados para permitir a visualização de um sistema sob diferentes perspectivas; nesse sentido um diagrama constitui uma projeção de um determinado sistema. A UML propõe os seguintes diagramas: *Diagrama de Casos de Uso*, *Diagrama de Classe*, *Diagrama de Objetos*, *Diagrama de Colaboração*, *Diagrama de Seqüência*, *Diagrama de Estado*, *Diagrama de Atividade*, *Diagrama de Implantação* e *Diagrama de Componente*.

A seguir são descritos sucintamente esses diagramas:

- *Diagrama de Casos de Uso*

Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema (sem ser necessário especificar como esse comportamento é implementado) e é uma descrição de um conjunto de seqüências de ações, incluindo variantes realizadas pelo sistema para produzir um resultado de valor observável por um ator (um ator representa um conjunto coerente de papéis que os usuários do sistema desempenham quando interagem com esses casos de uso).

Um modelo de caso de uso descreve os requisitos funcionais de um sistema em termos de casos de uso. Um diagrama de casos de uso mostra os relacionamentos entre os atores e casos de uso em interação com o sistema. Os diagramas de casos de uso são importantes para visualizar, especificar e documentar o comportamento de um elemento. Esses diagramas fazem com que sistemas, subsistemas e classes fiquem acessíveis e compreensíveis, por apresentarem uma visão externa sobre como esses elementos podem ser utilizados no contexto. Os diagramas de casos de uso também são importantes para testar sistemas executáveis por meio de engenharia de produção e para compreendê-los por meio de engenharia reversa [Booch 00].

- *Diagrama de Classe*

Diagrama que retrata uma série de elementos estáticos (essencialmente classes), juntamente com suas associações, estruturas de superclasse/subclasse e outros inter-relacionamentos estáticos. Opcionalmente, um diagrama desse tipo inclui, para cada classe, os seus atributos e operações [Booch 00].

- *Diagrama de Objeto*

Um diagrama de objetos exibe um conjunto de objetos e seus relacionamentos em um ponto do tempo. Representa retratos estáticos de instâncias de itens encontrados em diagramas de classe. São diagramas que abrangem a visão estática da estrutura ou do processo de um sistema, como ocorre nos diagramas de classes, mas sob perspectiva de casos reais ou de protótipos [Booch 00].

- *Diagrama de Colaboração*

Diagrama que mostra interações de objeto, ou seja as relações entre os objetos [Booch 00].

Diagramas de colaboração e seqüência (descrito a seguir) expressam informações semelhantes, mas as mostram de maneiras diferentes.

- *Diagrama de Seqüência*

Um diagrama que mostra interações de objeto organizados em uma seqüência de tempo, mostrando os objetos que participam na interação e a seqüência de mensagens trocadas. Ao contrário de um diagrama de colaboração, um diagrama de seqüência inclui seqüências de tempo mas não inclui relacionamentos de objeto. Pode existir em uma forma genérica (descrevendo todos os possíveis cenários) e em forma de instância (descrevendo um cenário real) [Booch 00].

- *Diagrama de Estado*

Diagrama que retrata estados (valores discretos de algum(ns) atributo(s)) e as transições permitidas entre esses estados. O diagrama de estado poderá também mostrar eventos (tais como mensagens de entrada para um objeto), que provocam transições, e mensagens de saída

provocadas por transições. São importantes principalmente para a modelagem de comportamentos de uma interface, classe ou colaboração e para dar ênfase a comportamentos de um objeto ordenados por eventos, o que é de grande ajuda na modelagem de sistemas reativos [Booch 00].

- *Diagrama de Atividade*

Um caso especial de diagrama de estado no qual todos ou a maioria dos estados são estados de ações e a maioria das transições é ativada por conclusão de ações no estados de origem. Abrange a visão dinâmica do sistema e é importante principalmente para a modelagem da função de um sistema e dá ênfase ao fluxo de controle entre objetos [Booch 00].

- *Diagrama de Implantação*

Diagrama que mostra a configuração de elementos de processando em tempo de execução e os componentes de software, processos, e objetos que se mantêm nele, ou seja, abrange a visão estática do funcionamento [Booch 00].

- *Diagrama de Componente*

Um diagrama que exhibe as organizações e as dependências existentes entre os componentes de software. Abrange a visão estática da implementação de um sistema [Booch 00].

3.5.1.2 – Método FUSION-RE/I Segundo a Notação UML

Como o método FUSION-RE/I utiliza modelos da fase de análise do método Fusion, foi primeiramente realizado uma pesquisa sobre sua adaptação à notação UML. Através do estudo de vários trabalhos encontrados na literatura sobre essa adaptação [TeamFusion 97][Atkinson 97][Sendall 99][Colanzi 99][Feltrim 99a] observou-se um consenso em relação aos diagramas UML correspondentes aos modelos produzidos pela fase de análise do método Fusion. A Tabela 3.6 mostra o Método FUSION-RE/I e quais artefatos produzidos por este método podem ser modelados segundo a notação UML [Feltrim 99a].

Tabela 3.6 – Método FUSION-RE/I x UML [Feltrim 99a]

MÉTODO FUSION-RE/I			
Etapa	Passo	Subpasso	Diagrama UML
1ª Etapa – Recuperar Visões Funcionais	Passo 1 – Obter Informações Existentes sobre o Sistema		-
	Passo 2 – Recuperar o Modelo de Análise do Sistema	Subpasso 2.1 – Elaborar o Modelo de Ciclo de Vida do Sistema	Diagrama de Estado
		Subpasso 2.2 – Elaborar o Modelo de Operações do Sistema	-

MÉTODO FUSION-RE/I (Continuação)			
Etapa	Passo	Subpasso	Diagrama UML
1ª Etapa – Recuperar Visões Funcionais	Passo 2 – Recuperar o Modelo de Análise do Sistema	Supasso 2.3 – Desenvolver o Modelo de Objetos do Sistema	Diagrama de Classe
2ª Etapa – Recuperar Visões Estruturais	Passo 1 – Elaborar o Quadro de Procedimentos de Implementação	Subpasso 1.1 – Elaborar o Quadro de Chamadas de Procedimentos	Diagrama de Componentes
		Subpasso 1.2 – Elaborar o Quadro de Índice de Procedimentos	-
	Passo 2 – Elaborar o Quadro das Operações-Procedimentos de Implementação		-

3.6 – Considerações Finais

Neste capítulo foi dada uma visão geral dos termos relacionados à tecnologia da engenharia reversa (níveis de abstração no ciclo de vida, definições de engenharia reversa, categorias de métodos de engenharia reversa) e apresentado o método de engenharia reversa orientado a objetos FUSION-RE/I e sua adaptação à notação UML.

No próximo capítulo será apresentada a abordagem SQUID, a qual envolve tanto a visão de qualidade como também a visão do processo de desenvolvimento de um produto de software e que será utilizada para o desenvolvimento deste trabalho.

Capítulo 4 – Abordagem SQUID

4.1 – Considerações Iniciais

O projeto *SQUID* (Software Quality In Development) iniciou em abril de 1994, dentro do programa ESPRIT da comunidade europeia. Está baseado nos conceitos e definições da norma ISO 9001, NBR 13596, ISO/IEC 14598-3 (Avaliação de Produto de Software – Parte 3: Processo para a Equipe de Desenvolvimento) e no modelo CMM. Alguns dos produtos do projeto *SQUID* são o método *SQUID* e a ferramenta *SQUID*, que apóia a aplicação desse método.

A seguir são apresentados o método *SQUID* e alguns detalhes da ferramenta associada, que aborda conjuntamente visões de produto e de processo de software e que será usado para o desenvolvimento desse trabalho de mestrado. Exemplos de aplicação da abordagem *SQUID* podem ser encontrados em Bøegh (no qual é descrito a aplicação da abordagem *SQUID* no gerenciamento e controle da qualidade do software durante o desenvolvimento de um sistema de controle e supervisão avançada projetado para desempenhar remotamente atividades experimentais na Antártida) [Bøegh 99] e Losavio (que descreve a aplicação da abordagem *SQUID* no gerenciamento e controle da qualidade do software em um processo de desenvolvimento orientado a objetos) [Losavio 99].

4.2 – A Abordagem SQUID para Qualidade

O objetivo do projeto *SQUID* é desenvolver e automatizar modelos e métodos criando um conjunto de ferramentas integradas, possibilitando que organizações que desenvolvem software possam monitorar, controlar e prever as qualidades de produtos nos vários estágios do ciclo de vida do desenvolvimento de software com o uso de medições objetivas [Bøegh 99].

As organizações usuárias desse conjunto de ferramentas devem ter definido um modelo de qualidade, um modelo de desenvolvimento de software, fazer medições dos projetos de software e ter um banco de dados com as medições de projetos anteriores da organização. Portanto, um usuário da abordagem *SQUID* deve definir um *Modelo de Qualidade* e seguir um *Processo de Qualidade*, que devem estar associados.

O **Modelo de Qualidade** deve abordar três visões [Bøegh 99]:

- Visão de Qualidade – define as características de qualidade desejadas do produto a serem medidas, que podem ser baseadas na NBR 13596;

- Visão de Dados – define os valores objetivos, valores atuais e valores estimados atribuídos às características definidas;
- Visão do Produto – define em quais fases do processo de desenvolvimento do produto serão realizadas as medições.

Com este *Modelo de Qualidade*, as organizações construiriam um Banco de Dados com:

- Informações sobre os critérios de qualidade utilizados – *Base de Conhecimento*;
- Informações sobre os dados colhidos de um projeto específico – *Base de Projeto*;
- Informações da execução dos projetos similares – *Base de Experiência*.

O **Processo de Qualidade SQUID** é um ciclo de quatro fases: *Especificação*, *Planejamento*, *Controle* e *Avaliação da Qualidade*. Tem como entrada as necessidades do usuário e como saída a qualidade em uso do produto (qualidade observável no produto de software segundo algum critério pré-estabelecido), como ilustrado na Figura 4.1.

Figura 4.1 – Fases do Processo de Qualidade – SQUID [Tsukumo 96]

Este método aborda as duas visões de avaliação de qualidade – produto e processo, com o enfoque no produto de software durante o seu desenvolvimento.

A abordagem SQUID propõe que se definam os requisitos de qualidade de produto de software através do estabelecimento de valores objetivos (*targets*) para as características de qualidade externas. Então buscam-se esses objetivos durante o processo de desenvolvimento definindo e monitorando os valores desejados através das características de qualidade internas. Isso pode ser feito usando-se medições de software convencionais relacionadas ao tamanho, taxa de erros, taxa de mudanças, estrutura, etc., obtidas nas fases de desenvolvimento do produto.

Para se estabelecer uma relação entre características de qualidade internas e externas, SQUID sugere que se use a experiência adquirida através de projetos anteriores, semelhantes. Experiências anteriores são usadas para avaliar a viabilidade dos requisitos de qualidade antes de se começar um projeto e para identificar e estabelecer os valores objetivos para as características de qualidade internas. Para que experiências anteriores sejam aproveitadas, há a necessidade que sejam armazenadas de uma maneira consistente, para permitir comparações através de diferentes projetos. SQUID usa um método rigoroso e organizado para identificação, definição, integração e comparação das medidas de qualidade através do Modelo de Dados SQUID (SQUID Data Model) e do Processo de Qualidade SQUID (SQUID Quality Process) [Bøegh 99].

4.3 – Modelo de Dados SQUID (SQUID Data Model)

Processos de desenvolvimento de software variam de empresa para empresa, e mesmo dentro de uma única empresa há normalmente diferentes modelos de desenvolvimento usuais. Essas diferenças afetam a possibilidade de uso de conjuntos de métricas através de projetos diferentes. Por exemplo, não é significativo comparar o número de linhas de código desenvolvido usando técnicas estruturadas com projetos desenvolvidos em um ambiente orientado a objetos. Além disso, diferentes processos afetam a relação entre medições e características de qualidade. Por exemplo, considerando-se a influência do "tamanho" (medido em termos de linhas de código) na manutenibilidade do software. Esta influência é diferente se o software é desenvolvido em um processo de desenvolvimento orientado a objeto ou em um modelo cascata [Bøegh 99]. Uma organização utilizando a abordagem SQUID terá que definir seu próprio **processo de desenvolvimento**, especificando os objetos de projeto produzidos por esse processo. Objetos de projeto representarão os elementos onde serão aplicadas as medições.

A abordagem SQUID requer que requisitos de qualidade sejam mapeados em características de qualidade e conseqüentemente em atributos de qualidade (propriedades mensuráveis) através de um **modelo de qualidade**, como por exemplo, o proposto pela NBR 13596. Requisitos de qualidade são então vistos como um conjunto de valores objetivos para as características de qualidade externas. Modelos de qualidade também identificam características de qualidade internas as quais influenciam as características externas [Bøegh 99]. Características de qualidade internas são então monitoradas e controladas através de **medições do projeto** com o objetivo de alcançar os requisitos de qualidade. O restante desta seção descreve esses três maiores componentes do Modelo de Dados SQUID: *Processo de Desenvolvimento*, *Modelo de Qualidade* e *Medições do Projeto*.

4.3.1 – Representando o Processo de Desenvolvimento

Já foi visto que as medições colecionadas durante um projeto de desenvolvimento de software são fortemente afetadas pelo processo de desenvolvimento. Medições precisam ser caracterizadas através da especificação do processo de desenvolvimento que foi utilizado. Isso é feito representando o processo de desenvolvimento no Modelo de Dados SQUID. O Modelo de Dados SQUID identifica as características essenciais do processo de desenvolvimento, tais como o tipo de modelo de desenvolvimento e a linguagem de programação adotada. Os elementos adicionais que são modelados são os objetos de projeto [Bøegh 99]. Os objetos de projeto são classificados em três diferentes tipos: *produtos* das fases de desenvolvimento do software (*deliverables*), *atividades de desenvolvimento* e *pontos de revisão* (Figura 4.2). Os *produtos* das fases de desenvolvimento do software são objetos de projeto produzidos durante o ciclo de vida de um projeto (especificação, código, etc.). *Atividades de desenvolvimento* produzem a liberação dos *produtos* das fases de desenvolvimento do software. *Atividades de desenvolvimento* e os *produtos* das fases de desenvolvimento do software são os elementos aos quais são aplicadas as medições de qualidade. *Pontos de revisão* são intervalos de controle usados em um modelo de desenvolvimento particular e durante as medições realizadas.

Figura 4.2 – Representando o Processo de Desenvolvimento [SQUID 98]

4.3.2 – Representando o Modelo de Qualidade

Um modelo de qualidade é especificado em termos de características de qualidade. Características de qualidade são refinadas até que sejam diretamente mensuráveis, e então são referenciadas como atributos de qualidade. Características e atributos de qualidade podem ser internos ou externos. O usuário identifica as características internas que influenciam as características externas criando um vínculo entre elas no Modelo de Dados SQUID [Bøegh 99]. Atributos de qualidade são propriedades do software diretamente mensuráveis que quantificam características de qualidade. A Figura 4.3 mostra como o modelo de qualidade SQUID é representado.

Figura 4.3 – Representando o Modelo de Qualidade [SQUID 98]

Segundo Bøegh [Bøegh 99], com exceção de projetos extremamente simples, partes diferentes de um produto de software têm normalmente requisitos comportamentais muito diferentes. Por exemplo, o componente de interface de um produto tem requisitos de usabilidade ao passo que componentes de banco de dados têm requisitos de integridade. O método SQUID, por essa razão usa o conceito de subsistema (*product portion*) para definir a parte de um produto que tem um distinto conjunto de requisitos de qualidade. Um único produto será composto de um conjunto de subsistemas. Os diferentes requisitos associados aos diferentes subsistemas são representados através de diferentes *valores objetivos* para as características de qualidade externas. O conceito de subsistema é um poderoso mecanismo para reduzir conflito entre requisitos de qualidade. Considerando que pode ser impossível obter requisitos operacionais extremamente altos, e às vezes conflitantes, para determinado produto como um todo, pode ser possível obterem-se requisitos de qualidade restritos a subsistemas do produto, uma vez que os requisitos são distribuídos aos subsistemas que genuinamente os requerem. A Figura 4.4 ilustra como são representados os atributos e o modelo de qualidade na ferramenta SQUID.

Figura 4.4 – Representando Atributos e o Modelo de Qualidade na Ferramenta SQUID

4.3.3 – Representando Medidas

As medidas coletadas pela ferramenta SQUID podem ser de três tipos: *correntes*, *estimativas*, ou *desejadas (targets)*. *Medida corrente* é o valor medido por um atributo de qualidade específico em um determinado momento do desenvolvimento. *Medida estimada* é uma estimativa do valor, que pode ser obtido através de dados armazenados anteriormente ou pela facilidade de estimação provida pela ferramenta SQUID. *Medida desejada* é o valor objetivo especificado durante a fase de especificação da qualidade. Medições podem ser expressas em unidades diferentes. Por exemplo, linhas de código é uma unidade para tamanho do código fonte do produto, dias decorridos é uma unidade para duração do projeto, e anos de trabalho é uma unidade para experiência pessoal. O usuário da ferramenta SQUID define, nas regras de contagem (*counting rules*), os procedimentos a serem seguidos para se fazerem as medições. Uma *regra de contagem* é uma descrição de como uma medição deve ser feita, esclarece e regulamenta todas as condições que podem afetar a objetividade da medição. Por exemplo, uma medição do tamanho do programa pode ser coletada em termos de linhas de código, mas a menos que uma regra de contagem confirme em qual estágio de desenvolvimento a medição deve ser feita, o usuário pode obter valores não significativos. Regras de contagem são necessárias para assegurar que medições sejam repetíveis e que sejam comparáveis com valores armazenados no banco de dados histórico [Bøegh 99]. A Figura 4.5 ilustra como as medidas são representadas na abordagem SQUID.

Figura 4.5 – Representando Medidas [SQUID 98]

4.3.4 – Integrando Componentes do Modelo de Dados SQUID

Todos os atributos internos e externos pertencentes a um **modelo de qualidade** específico devem ser alocados aos tipos de objetos do projeto, os quais pertencem a um específico **processo de desenvolvimento** e devem ser associados a unidades e regras de contagens (**medições do**

projeto) no modelo de dados SQUID. Assim, se o atributo "complexidade estrutural" é parte do modelo de qualidade, este atributo deve ser associado a um apropriado tipo de objeto de projeto no modelo de processo de desenvolvimento, por exemplo, um módulo de desenvolvimento em cascata. A organização resultante do modelo de dados SQUID é mostrada na Figura 4.6.

Figura 4.6 – Representação Simplificada do Modelo de Dados SQUID [Bøegh 99]

4.4 – O Processo de Qualidade SQUID (SQUID Quality Process)

O Processo de Qualidade SQUID inclui: *Especificação da Qualidade*; *Planejamento e Controle da Qualidade* e *Avaliação da qualidade*. Esta seção descreve como a ferramenta SQUID interage e apóia o usuário durante o processo de qualidade.

4.4.1 – Especificação da Qualidade

A *Especificação da Qualidade* é responsável por estabelecer os requisitos de qualidade do produto de software. As principais atividades da *Especificação da Qualidade* são [Bøegh 99]:

- Selecionar o modelo de qualidade e o processo de desenvolvimento do software. As especificações dos requisitos de qualidade são baseadas em uma estrutura de algum modelo de qualidade específico (como descrito na seção 4.3). Qualquer organização que utilize a abordagem SQUID necessita definir modelos de qualidade, usados para desenvolver seus produtos de software. Estes modelos poderiam ser baseados em um padrão como a NBR 13596 ou qualquer outro modelo definido ou utilizado pela empresa. Quando um novo projeto é iniciado, a organização tem que determinar quais dos modelos de qualidade disponíveis e qual o processo de desenvolvimento deve-se aplicar ao projeto atual.

- Identificar subsistemas, requisitos de qualidade e os valores objetivos para os atributos de qualidade. O produto que é desenvolvido deve ser definido em termos de um conjunto de subsistemas independentes. Subsistemas são partes de um produto de software que possui requisitos de qualidade diferentes. Então, cada requisito operacional é associado com o subsistema ao qual se aplica. A ferramenta SQUID auxilia a associação de cada requisito com uma característica de qualidade, conforme o modelo de qualidade selecionado. Uma vez que uma associação é feita entre uma característica de qualidade e um requisito, a ferramenta identifica um ou mais atributos externos, derivados do modelo de qualidade, nos quais os valores objetivos podem ser estabelecidos. A Figura 4.7 ilustra a introdução de requisitos de qualidade na ferramenta SQUID.

Figura 4.7 – Especificando os Requisitos de Qualidade (Visão Externa)

- Desempenhar a análise da viabilidade. A ferramenta SQUID permite avaliar a viabilidade dos valores estabelecidos para os atributos externos. Avaliação da viabilidade está baseada na comparação de projetos similares realizados anteriormente com sucesso, com os valores objetivos do projeto corrente. A ferramenta SQUID analisa a base de dados histórica procurando por projetos similares. O nível de similaridade é julgado pela análise do processo de desenvolvimento usado por projetos anteriores e seus valores requeridos para os atributos externos. A ferramenta SQUID então analisa os valores atuais dos atributos externos obtidos pelos projetos anteriores selecionados. Estes valores são comparados com valores objetivos do novo produto. Se os valores atuais são diferentes aos valores objetivos, há um risco que o processo de desenvolvimento usual não possa entregar um produto que satisfaça os requisitos de qualidade declarados. É claro que este aspecto requer a presença de uma base de dados histórica volumosa. Nenhum auxílio pode ser provido se esta base de dados não estiver disponível visto que a filosofia da abordagem SQUID é que medições devem ser definidas no contexto específico de uma organização [Bøegh 99].

4.4.2 – Planejamento e Controle da Qualidade

O progresso da qualidade é monitorado ao longo do desenvolvimento, utilizando-se medições internas do software associado com medições dos produtos (das fases de desenvolvimento do software) e atividades de desenvolvimento relacionadas, em cada ponto principal de revisão no desenvolvimento [Kitchenham 95]. As principais atividades do planejamento e controle da qualidade são [Bøegh 99]:

- Associar os valores objetivos aos atributos de qualidade internos, com pontos de revisão definidos. Objetivos quantitativos são definidos para os atributos de qualidade internos, identificados no modelo de qualidade. Atributos são então associados a atividades intermediárias e aos produtos (objetos de projeto), junto ao qual o progresso será monitorado e controlado. A ferramenta SQUID fornece assistência para se associar os valores objetivos às medidas internas, permitindo ao usuário observar os valores dos atributos de qualidade que foram alcançados em projetos semelhantes. A Figura 4.8 ilustra a instanciação de objetos de projeto e a definição de valores objetivos.

Figura 4.8 – Planejamento da Qualidade (Visão Interna)

- Identificar divergências pela comparação dos valores objetivos com os valores atuais dos atributos de qualidade internos. A cada ponto de revisão do projeto, quando determinado atributo de qualidade está sendo medido, o *status* da qualidade atual do projeto pode ser avaliado. Isto envolve a comparação de cada atributo de qualidade interno com seu valor requerido e a identificação de qualquer desvio significativo. Dentro do contexto de um modelo de qualidade de software que relaciona atributos internos e externos, isto ajuda a avaliar se é provável que o projeto alcance os valores objetivos dos atributos de qualidade externos.
- Identificar componentes do software com valores de atributos incomuns. Estes componentes devem ser analisados mais cuidadosamente para assegurar uma identificação antecipada de áreas problemáticas. A ferramenta SQUID provê um método de automatizar a detecção de componentes anômalos para identificar objetos de projeto com combinações de valores de

atributos internos incomuns. A Figura 4.9 ilustra o recurso de detecção de componentes anômalos fornecido pela ferramenta SQUID.

Figura 4.9 – Identificando Componentes Anômalos

4.4.3 – Avaliação da Qualidade

A atividade de *Avaliação da Qualidade* se preocupa em assegurar que um produto satisfaça seus requisitos especificados. Durante a avaliação da qualidade, a equipe de desenvolvimento pode também analisar o resultado do projeto para prover um *feedback* para o processo de desenvolvimento. O método e a ferramenta SQUID provêem um mecanismo básico de avaliação baseado em [Bøegh 99]:

- Medir os valores atuais dos atributos externos para cada subsistema;
- Comparar o valor atual com o valor desejado;
- Relatar falhas de qualidade.

A Figura 4.10 mostra a seqüências das atividades do *Processo de Qualidade SQUID* e suas principais relações com a base de dados.

Figura 4.10 – Atividades do Processo de Qualidade e Relações com a Base de Dados SQUID [Baegh 99]

4.5 – Considerações Finais

Neste capítulo foram introduzidos os conceitos relativos à abordagem SQUID e alguns detalhes da ferramenta SQUID (modelo de desenvolvimento, modelo de qualidade, medições de projeto, integração dos modelos de qualidade e do modelo de desenvolvimento, e processo de qualidade SQUID).

Os Capítulos 2, 3 e 4 serviram de base para bom estudo e entendimento deste trabalho o qual é proposto no capítulo a seguir, apresentando a modelagem de um processo de engenharia reversa com base no método FUSION E/I segundo a abordagem SQUID.

Capítulo 5 – Processo de Engenharia Reversa Segundo a Abordagem SQUID

5.1 – Considerações Iniciais

Engenharia reversa vem se tornando um campo de pesquisa crescente e popular dentro da engenharia de software. Segundo Bennett [Bennett 98] há poucos trabalhos publicados sobre como abordagens de engenharia de reversa podem ser avaliadas e comparadas. Isto torna difícil apresentar um caso de aplicação industrial convincente. Talvez esse fato explique porque a engenharia reversa não seja amplamente utilizada nas empresas de software. Há a necessidade de mecanismos que auxilie no planejamento e controle de sua aplicação.

Este capítulo descreve a modelagem de um método de engenharia reversa (FUSION-RE/I) segundo a abordagem SQUID, a qual permitirá a obtenção de algumas medidas da qualidade atual do produto e dos artefatos obtidos durante o processo de engenharia reversa, auxiliando assim na avaliação da qualidade do produto submetido à engenharia reversa e do próprio processo de engenharia reversa.

5.2 – Modelo de Desenvolvimento do Processo de Engenharia Reversa

Um modelo de desenvolvimento segundo a abordagem SQUID deve representar os métodos de desenvolvimento de software utilizados por uma empresa e envolve a definição de três tipos de objetos pertencentes ao processo:

- *Pontos de Revisão*, que são pontos de controle utilizado em um modelo de desenvolvimento particular;
- *Atividades de Desenvolvimento*, que são os objetos de projeto os quais produzem os produtos (*deliverables*);
- *Produtos*, que são os tipos de objetos do projeto¹ (artefatos) produzidos durante o ciclo de vida de desenvolvimento de um projeto (ou subprojeto).

¹ Neste contexto Projeto tem o significado da aplicação do processo de engenharia (progressivo/reversa) a um sistema.

A finalidade deste tipo de representação é permitir associar diferentes medidas internas do produto em desenvolvimento com os diferentes pontos de revisão ou comparar o estado de um projeto novo com o estado de projetos anteriores em um mesmo ponto de revisão.

Quando se define um modelo de desenvolvimento deve-se especificar cada tipo de objeto de projeto como *repetível* ou *não repetível*, indicando a existência ou não de várias instâncias do mesmo tipo de objeto.

A seguir são descritos os pontos de revisão, atividades de desenvolvimento e os produtos gerados por essas atividades, propostos para o método FUSION-RE/I.

Pontos de Revisão (Modelo de Desenvolvimento)

Na Tabela 5.1 são indicados os pontos de revisão identificados no processo de engenharia reversa, os quais nesta modelagem correspondem às etapas fundamentais do método de engenharia reversa FUSION-RE/I (descritas no Capítulo 3), exceto pelas seguintes modificações:

- Ponto de revisão *Recuperação de informações*. A recuperação de informações pode estender-se durante várias etapas da aplicação do método de engenharia reversa FUSION-RE/I. Uma vez que o método não define diretrizes específicas para esta etapa, ela foi separada da etapa de *Recuperação de visões funcionais*.
- Ponto de revisão *Gerenciamento do projeto*. Incluído para controlar informações de um projeto de engenharia reversa e seu escopo é determinado pelo tempo efetivo de projeto.

Para efeito de identificar etapas, passos, subpassos e produtos gerados pelo processo de engenharia reversa, utiliza-se uma numeração na modelagem dos tipos de objetos de projeto, não implicando necessariamente uma ordem rígida (algumas atividades de desenvolvimento podem ocorrer paralelamente a outras).

Tabela 5.1 – Pontos de Revisão

Pontos de Revisão	Tipo	Descrição
1. Recuperação de informações	NR	Final da fase de recuperação de informações existentes sobre o sistema.
2. Recuperação de visões funcionais	NR	Final da fase de recuperação de visões funcionais do sistema.
3. Recuperação de visões estruturais	NR	Final da fase de recuperação de visões estruturais do sistema.
4. Gerenciamento do projeto	NR	Final da fase de gerenciamento do projeto de engenharia reversa.
Legenda: NR = Não Repetível		

Cada ponto de revisão indica o final de uma fase específica, como especificado no método de engenharia reversa e portanto definido como não repetível. Neste contexto significa dizer que o

modelo de desenvolvimento (para a engenharia reversa) segue um modelo de ciclo de vida em cascata.

Atividades de Desenvolvimento (Modelo de Desenvolvimento)

Na Tabela 5.2 são especificados as atividades de desenvolvimento identificadas no processo de engenharia reversa utilizando o método FUSION-RE/I.

As atividades de desenvolvimento foram consideradas não repetíveis porque o método FUSION-RE/I define essa sequência de atividades como únicas (as quais por sua vez podem gerar um ou mais produtos).

Tabela 5.2 – Atividades de Desenvolvimento

Atividades	Tipo	Descrição
1.1 Obter informações existentes sobre o sistema	NR	Obter Informações disponíveis do sistema.
2.1 Elaborar modelo de ciclo de vida do sistema	NR	Elaborar o modelo de ciclo de vida do sistema a partir da interface do sistema.
2.2 Elaborar modelo de operações do sistema	NR	Elaborar o modelo de operações do sistema através do uso intensivo do sistema.
2.3 Definir temas	NR	Definir temas relacionados com a funcionalidade do sistema.
2.4 Elaborar modelo de objetos do sistema	NR	Elaborar o modelo de objetos para cada tema definido.
3.1 Elaborar quadro de chamadas de procedimentos	NR	Elaborar um quadro contendo todos os procedimentos contidos nos arquivos, fazendo-se referência aos procedimentos utilizados (chama) e utilizadores (chamado por).
3.2 Elaborar quadro de índice dos procedimentos	NR	Elaborar um quadro (em ordem alfabética) contendo todos os procedimentos de implementação do sistema.
3.3 Elaborar quadro de operações – procedimentos de implementação	NR	Elaborar um quadro que associa cada procedimento do quadro de chamadas às operações da interface do sistema e aloca-los, de acordo com sua funcionalidade , à interface ou a um dos temas do sistema.
4.1 Obter informações sobre o gerenciamento do projeto	NR	Obter informações sobre o gerenciamento do projeto de engenharia reversa.
Legenda: NR = Não Repetível		

Produtos (Modelo de Desenvolvimento)

A Tabela 5.3 apresenta os produtos gerados pelas atividades de desenvolvimento.

Na modelagem SQUID, a atividade *Elaborar quadro de operações-procedimentos de implementação* foi definida para gerar dois produtos:

- Quadro de operações-procedimentos de implementação (interface), definido como não repetível porque está relacionado somente a uma instância (interface);

- Quadro de operações-procedimentos de implementação (temas), definido como repetível para permitir que cada quadro esteja relacionado a um tema e assim ser possível obter informações individuais de cada quadro.

Um sistema submetido ao processo de engenharia reversa pode possuir vários módulos e temas definidos, então os produtos *Informações sobre o sistema (módulos)* e *Temas do sistema* foram definidos como repetíveis, assim permitindo coletar métricas para cada um desses produtos.

O método FUSION-RE/I recomenda que se crie um quadro de chamadas de procedimentos para cada módulo do sistema. Na modelagem realizada, o produto *Quadro de chamadas de procedimentos* foi definido como não repetível, mas representa o conjunto de todos os quadros obtidos, isso porque as métricas selecionadas a esse produto são referentes aos procedimentos e não aos módulos a que pertencem.

Os demais itens foram identificados como produtos de uma única instância em cada projeto, devido ao fato que suas respectivas atividades de desenvolvimento segundo o método FUSION-RE/I os definem como produtos únicos.

Tabela 5.3 – Produtos Gerados pelas Atividades de Desenvolvimento

Produtos	Tipo	Descrição
1.1 Informações sobre o sistema (executável)	NR	Informações obtidas sobre o sistema executável.
1.2 Informações sobre o sistema (código fonte)	NR	Informações obtidas sobre o sistema através do código fonte.
1.3 Informações sobre o sistema (módulos)	R	Informações sobre os módulos que compõem o sistema.
2.1 Modelo de ciclo de vida	NR	Modelo de ciclo de vida do sistema.
2.2 Modelo de operações	NR	Modelo de operações do sistema.
2.3 Temas do sistema	R	Temas do sistema.
2.4 Modelo de objetos	NR	Modelo de objetos do sistema (engloba o modelo de objetos de todos os temas definidos para o sistema)
3.1 Quadro de chamadas de procedimentos	NR	Quadro contendo todos os procedimentos e a relação chama/chamado por.
3.2 Quadro de índice de procedimentos	NR	Quadro contendo todos os procedimentos (alfabeticamente).
3.3.1 Quadro de operações-procedimentos de implementação (interface)	NR	Quadro de operações-procedimentos de implementação referente à interface.
3.3.2 Quadro de operações-procedimentos de implementação (temas)	R	Quadro de operações-procedimentos de implementação referente aos temas do sistema.
4. Informações sobre o gerenciamento do projeto	NR	Informações sobre o gerenciamento do projeto de engenharia reversa.
Legenda: Repetível = R; Não Repetível = NR		

5.3 – Modelo de Qualidade do Processo de Engenharia Reversa

Uma vez definido o modelo de desenvolvimento é necessário definir o seu modelo de qualidade associado, isto é, estabelecer quais são as características que se espera do produto desse processo para ser considerado um produto de qualidade. Como se trata de um processo de engenharia reversa, onde o objetivo é obter informações do sistema e recuperar o modelo de análise do sistema visando a auxiliar na manutenção ou em uma reengenharia, foram identificados quatro características de qualidade relevantes ao processo: *Manutenibilidade*, *Portabilidade*, *Usabilidade* e *Projeto*. A escolha dessas características foi realizada após o estudo dos produtos gerados pelo método FUSION-RE/I, da identificação de seus respectivos atributos de qualidade e também de alguns atributos de qualidade que poderiam ser obtidos através do uso do sistema submetido à engenharia reversa. Depois de identificados esses atributos de qualidade foi realizado um mapeamento para associá-los às características de qualidade. Neste trabalho foi utilizado para esse fim a norma NBR 13596 [NBR 13596] por ser um modelo de qualidade adotado internacionalmente. A característica *Projeto* foi adicionada ao modelo visando coletar informações gerenciais do próprio projeto de engenharia reversa como data de início do projeto, número de pessoas alocadas ao projeto e quais atividades do processo de engenharia reversa já foram executadas.

A Figura 5.1 mostra as características e subcaracterísticas que foram selecionadas com base no estudo dos padrões de qualidade para o produto de software (descritos no Capítulo 2) para se definir o modelo de qualidade proposto.

Figura 5.1 – Modelo de Qualidade Proposto

5.4 – Seleção de Métricas

Métricas são utilizadas para permitir a quantificação do grau em que os atributos estão presentes em um determinado produto de software de acordo com um modelo de medição especificado. Fenton [Fenton 97] define medição como o processo pelo qual números ou símbolos são atribuídos aos atributos de entidades de tal forma que as descrevam de acordo com regras claramente definidas. Para que seja possível a medição, portanto, devemos conhecer quais as entidades que estão sendo analisadas e os atributos da entidade que se desejam quantificar. Entendemos, portanto, métricas como medidas numéricas ou simbólicas que servem para indicar a adequação do software às características de qualidade às quais estão relacionadas [Shepperd 92][Fenton 97].

Para a obtenção de bons resultados, o processo de medição deve ser implantado de forma eficaz. Alguns princípios úteis a serem adotados na implantação de processo de medição são [Jones 97]:

- Medições devem ser usadas para medir processos não pessoas;
- O processo de medição deve ter objetivos claros e bem-definidos;
- O processo de medição deve ser integrado com o processo de desenvolvimento de software;
- O processo de medição deve ser fortemente acoplado com o processo de gerência da qualidade e integrado dentro de planos e orçamentos;

- O processo de coleta de dados deve ser simples, e ferramentas automáticas para extração de dados devem ser usadas;
- As medições devem ser repetíveis e independente do observador;
- O processo de medição é um processo contínuo e sujeito a melhoria;
- Os resultados devem ser compartilhados com as equipes de desenvolvimento de software;
- O processo de medição deve quantificar a qualidade e a produtividade do processo, bem como medir o impacto de ferramentas, métodos e técnicas de melhoria da qualidade e de produtividade do processo, fornecendo uma base quantitativa para a avaliação das iniciativas de melhoria do processo, e conseqüentemente, promovendo sua melhoria contínua.

As métricas selecionadas para comporem o modelo de qualidade foram definidas segundo os seguintes critérios:

- Prover resultados consistentes;
- Ser de fácil aprendizado;
- Servir para estimativas;
- Possibilitar obter séries históricas.

Outro critério adotado foi permitir sua fácil obtenção pensando nas micro e pequenas empresas de desenvolvimento de software nacionais que na sua maioria se encontram em um baixo nível de maturidade e que ainda não possuem uma cultura voltada à qualidade de software e coleta de métricas (como descrito na seção 2.7 do Capítulo 2). Neste aspecto este trabalho também busca a introdução dessa política de coleta de métricas no dia-a-dia dessas empresas, de forma que essas empresas possam criar assim uma base histórica de métricas que auxilie no planejamento de projetos futuros.

Como o método FUSION-RE/I recupera o modelo de análise de um sistema usando uma abordagem orientada a objetos de sistemas que foram implementados sem usar essa tecnologia, foram identificados métricas referentes ao sistema legado (procedimentais), aos produtos do processo (orientadas a objetos) e outras referentes ao próprio processo. Essa classificação é ilustrada na Figura 5.2.

Figura 5.2 – Tipos de Métricas do Modelo de Qualidade

5.4.1 – Métricas Externas

Segundo as normas NBR 13596 [NBR 13596], ISO/IEC 9126-3 [ISO/IEC 9126-3], as métricas externas (no modelo de qualidade SQUID referenciadas como atributos externos) devem ser projetadas para descrever a magnitude na qual o software satisfará as necessidades implícitas ou estabelecidas por parte do usuário durante a operação do sistema. Para isso se deve medir o comportamento do software em teste ou operação, a integração com outros softwares, hardware ou sistema e o comportamento do usuário. Em um projeto de engenharia reversa os objetos de estudo são: o próprio sistema executável, sua documentação existente (incluindo o código fonte) e os artefatos produzidos durante o processo de engenharia reversa. Nesta perspectiva pode-se considerar o tempo decorrido na análise e obtenção dos mesmos como atributos externos. Na Tabela 5.4 é especificada a visão externa da qualidade para o modelo proposto.

Tabela 5.4 – Atributos Externos

Atributo Externo	Descrição
Tempo na análise do código	Tempo na análise do código fonte de todos os módulos que compõem o sistema.
Tempo na definição dos temas	Tempo na definição dos temas do sistema.
Tempo na elaboração do modelo de ciclo de vida	Tempo na elaboração do modelo de ciclo de vida do sistema.
Tempo na elaboração do modelo de objetos	Tempo na elaboração do modelo de objetos (de todos os temas do sistema).
Tempo na elaboração do quadro de índice de procedimentos	Tempo na elaboração do quadro de índice de procedimentos do sistema.

Atributo Externo (Continuação)	Descrição
Tempo na elaboração do quadro de operações-procedimentos de implementação	Tempo na elaboração do quadro de operações-procedimentos de implementação do sistema (interface e temas do sistema).
Tempo na elaboração do quadro de chamadas de procedimentos	Tempo na elaboração do quadro de chamadas de procedimentos do sistema.
Tempo na coleta de informações	Tempo na coleta de informações sobre o domínio da aplicação, entrevistas com usuários e desenvolvedores, etc.
Tempo no entendimento das funcionalidades do sistema	Tempo para entender as funcionalidades do sistema.
Tempo de projeto	Tempo na engenharia reversa do sistema.

A Tabela 5.5 descreve as características e subcaracterísticas externas propostas para modelo de qualidade.

Tabela 5.5 – Descrição das Características e Subcaracterísticas Externas

Características	Subcaracterísticas	Descrição
Manutenibilidade(E)		Conjunto de atributos que evidenciam o esforço necessário para realizar modificações específicas no software [NBR 13596].
Manutenibilidade(E)	Analísabilidade(E)	Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas [NBR 13596].
Projeto(E)		Atributos relacionados ao gerenciamento do projeto de engenharia reversa.

Para compor a visão externa do modelo de qualidade é necessário associar cada atributo externo a uma característica ou subcaracterística externa do modelo de qualidade. Essa associação é mostrada na Tabela 5.6.

Tabela 5.6 – Visão Externa da Qualidade

Característica	Subcaracterística	Atributo Externo
Manutenibilidade(E)	Analísabilidade(E)	Tempo no entendimento das funcionalidades do sistema.
		Tempo na coleta de informações
		Tempo na análise do código.
		Tempo na elaboração do modelo de ciclo de vida.
		Tempo na definição dos temas.
		Tempo na elaboração do modelo de objetos.
		Tempo na elaboração do quadro de índice de procedimentos.
		Tempo na elaboração do quadro de chamadas de procedimentos.
		Tempo na elaboração do quadro de operações-procedimentos de implementação.
Projeto(E)		Tempo de projeto

5.4.2 – Métricas Internas

As métricas internas (no modelo de qualidade SQUID referenciadas como atributos internos) permitem avaliar a qualidade de produtos intermediários durante o desenvolvimento deste, com o objetivo de verificar se os produtos de software intermediários e finais satisfazem os requisitos de qualidade internos, planejados para melhorar a qualidade de projetos, programas ou processos [Losavio 99]. Em geral, avaliam a estrutura interna, relações e funcionalidades de componentes de software.

No caso do desenvolvimento progressivo o objetivo é identificar atributos internos dos artefatos produzidos durante o desenvolvimento do software que de alguma forma influenciam a qualidade externa do sistema final. Num processo de engenharia reversa, os atributos internos podem ser relacionados às informações obtidas sobre o sistema executável, sobre o código fonte, sobre os artefatos produzidos pelo processo e informações do próprio processo de engenharia reversa. A Tabela 5.7 descreve os atributos internos identificados na modelagem do processo de engenharia reversa do método FUSION-RE/I. Para o produto *Modelo de ciclo de vida* foram propostos os atributos internos *Número de sentenças* (caso esteja representado na notação do método Fusion), *Número de estados* e *Número de transições* (caso esteja representado na forma de um diagrama de estados da notação UML). As demais adaptações descritas no Capítulo 3 ao método FUSION-RE/I para apoiar a notação UML (produtos: *Modelo de objetos* e *Quadro de chamadas de procedimentos*) não são afetadas pelos atributos internos selecionados, unidades de medida ou regras de contagens associadas porque estes independem da notação adotada.

Tabela 5.7 – Atributos Internos

Atributo Interno	Descrição
Atividade executada	Indica se uma determinada atividade já foi executada. Esse atributo interno foi necessário devido ao fato que na ferramenta, cada atividade de desenvolvimento deve ter pelo menos um atributo interno associado para se poder realizar a integração.
Classes (modelo de objetos)	Total de classes do modelo de objetos (referentes a todos os temas que compõem o sistema).
Classes (tema)	Número de classes de um tema do sistema.
Data de início do projeto	Data do início do projeto de engenharia reversa do sistema em questão.
Data final da fase de recuperação de informações	Data final da fase de recuperação de informações sobre o sistema.
Data final da fase de recuperação de visões estruturais	Data final da fase de recuperação de visões estruturais do sistema.
Data final da fase de recuperação de visões funcionais	Data final da fase de recuperação de visões funcionais do sistema.
Data final do projeto	Data final do projeto de engenharia reversa do sistema
Densidade de comentários (módulo)	Razão entre linhas de comentários e linhas executáveis do código fonte de um módulo.
Densidade de comentários (sistema)	Razão entre linhas de comentários e linhas executáveis do código fonte do sistema (de todos os módulos que compõem o sistema).

Atributo Interno :	Descrição (continuação)
Densidade de módulos dependentes de hardware	Razão entre número de módulos dependentes de hardware e número de módulos do sistema.
Densidade de procedimentos (interface)	Razão entre número de procedimentos associados à interface e número de procedimentos do sistema.
Densidade de procedimentos (tema)	Razão entre número de procedimentos associados a um tema e número de procedimentos do sistema.
Fan-in	Número de módulos chamando um módulo.
Fan-in (procedimentos)	Número de procedimentos chamando um procedimento.
Fan-out	Número de módulos chamados por um módulo.
Fan-out (procedimentos)	Número de procedimentos que são chamados por um procedimento.
Linguagem de implementação	Linguagem de implementação de um módulo do sistema.
Linhas de comentários (módulo)	Linhas de comentários de um módulo do sistema.
Linhas de comentários (sistema)	Total de linhas de comentários dos módulos que compõem o sistema.
Módulos dependentes de hardware	Número de módulos que contém implementações dependentes do hardware.
Módulos do sistema	Número total de módulos do sistema.
Nível de conhecimento	Nível de conhecimento do pessoal responsável pela engenharia em relação às linguagens de implementação dos módulos do sistema.
Número de estados	Número de estados de um diagrama de estados.
Número de operações	Número de operações do modelo de operações.
Número de sentenças	Número de sentenças do modelo de ciclo de vida do sistema (notação Fusion)
Número de transições	Número de transições de um diagrama de estados.
Opções de menu (total)	Número total de opções do menu do sistema.
Pessoas alocadas ao projeto	Número de pessoas responsáveis pela engenharia reversa do sistema.
Procedimentos (interface)	Número de procedimentos identificados como implementação da interface do sistema.
Procedimentos (módulo)	Número de procedimentos implementados em um módulo.
Procedimentos (sistema)	Número total de procedimentos implementados no sistema.
Procedimentos (tema)	Número de procedimentos identificados como implementação do tema.
Profundidade menu	Opções requeridas para desempenhar uma função (máximo).
Tamanho do módulo (loc executáveis)	Número de linhas de código executáveis do módulo.
Tamanho do módulo (loc)	Número de linhas de código de um módulo do sistema.
Tamanho do sistema (loc executáveis)	Número de linhas de código executáveis do sistema.
Tamanho do sistema (loc)	Número de linhas de código do sistema.
Telas do sistema	Número total de telas do sistema.
Tipo de plataforma	Plataforma operacional para a qual o sistema foi desenvolvido.

A Tabela 5.8 descreve as características e subcaracterísticas internas propostas para o modelo de qualidade.

Tabela 5.8 – Descrição das Características e Subcaracterísticas Internas

Características	Subcaracterísticas	Subcaracterísticas	Descrição
Manutenibilidade(I)			Conjunto de atributos que evidenciam o esforço necessário para realizar modificações específicas no software [NBR 13596].
Manutenibilidade(I)	Modularidade		O grau para o qual um sistema ou programa é composto de componentes discretos tal que uma mudança em um componente tem impacto mínimo sobre outros componentes.
Manutenibilidade(I)	Modularidade	Acoplamento	O grau de interdependência entre os módulos do software.
Manutenibilidade(I)	Modularidade	Simplicidade	O grau para o qual um sistema ou componente tem um projeto ou implementação que é direto e facilmente entendido.
Manutenibilidade(I)	Analísabilidade(I)		Atributos do software que evidenciam o esforço necessário para diagnosticar deficiências ou causas de falhas, ou para identificar partes a serem modificadas [NBR 13596].
Manutenibilidade(I)	Analísabilidade(I)	Legibilidade do código	O grau no qual as funções do software e sua implementação podem ser facilmente entendidas através da análise do código fonte.
Projeto(I)			Atributos relacionados ao gerenciamento do projeto de engenharia reversa.
Usabilidade(I)			Conjunto de atributos que evidenciam o esforço necessário para se poder utilizar o software [NBR 13596].
Usabilidade(I)	Operacionalidade		Atributos do software que evidenciam o esforço do usuário para sua operação e controle da sua operação [NBR 13596].
Portabilidade(I)			Conjunto de atributos que evidenciam a capacidade do produto de software ser transferido de um ambiente para outro [NBR 13596].
Portabilidade(I)	Adaptabilidade		Atributos do software que evidenciam sua capacidade de ser adaptado a ambientes diferentes especificados, sem a necessidade de aplicação de outras ações ou meios além daqueles fornecidos para essa finalidade pelo software considerado [NBR 13596].

Como no caso da visão externa do modelo de qualidade, é necessário associar cada atributo interno a uma característica ou subcaracterística interna do modelo de qualidade proposto. Essa associação é mostrada na Tabela 5.9.

Tabela 5.9 – Visão Interna da Qualidade

Característica	Subcaracterística	Subcaracterística	Atributo Interno
Manutenibilidade(I)	Modularidade	Acoplamento	Fan-in
			Fan-out
			Fan-in (procedimentos)
			Fan-out (procedimentos)
		Simplicidade	Tamanho sistema (loc)
			Tamanho do sistema (loc executáveis)
			Tamanho do módulo (loc)
			Tamanho do módulo (loc executáveis)
			Módulos do sistema
			Procedimentos (módulo)
			Procedimentos (sistema)
			Classes (modelo de objetos)
			Classes (tema)
			Número de sentenças
			Número de estados
			Número de transições
			Número de operações
			Procedimentos (interface)
			Procedimentos (tema)
			Densidade de procedimentos (interface)
			Densidade de procedimentos (tema)
	Analisabilidade(I)	Legibilidade do código	Densidade de comentários (módulo)
			Densidade de comentários (sistema)
			Linhas de comentários (sistema)
			Linhas de comentários (módulo)
Projeto(I)			Pessoas alocadas ao projeto
			Nível de conhecimento
			Data de início do projeto
			Data final do projeto
			Data final da fase de recuperação de informações
			Data final da fase de recuperação de visões funcionais
			Data final da fase de recuperação de visões estruturais
			Atividade executada

Característica	Subcaracterística	Subcaracterística	Atributo Interno (Continuação)
Usabilidade(I)	Operacionalidade		Profundidade menu
			Opções de menu (total)
			Telas do sistema
Portabilidade(I)	Adaptabilidade		Linguagem de implementação
			Módulos dependentes de hardware
			Densidade de módulos dependentes de hardware
			Tipo de plataforma

Para cada atributo externo ou interno mensurável, deve-se estabelecer uma unidade de medida. Por outro lado, para que as medidas sejam repetíveis e comparáveis deve-se definir regras de contagem, que definem procedimentos e condições sobre as quais uma medida é obtida.

A Tabela 5.10 descreve todas as unidades de medidas utilizadas pelos atributos de qualidade definidos, assim como seu tipo de escala e intervalos válidos. Os intervalos de valores válidos dos tipos inteiro e real foram superdimensionados para que não haja restrições nas entradas desses valores devido ao tamanho sistema original, forma de implementação, produtos do processo de engenharia reversa, etc.

Tabela 5.10 – Definição de Unidades de Medida

Unidade	Tipo Escala /Valores Válidos	Descrição
Módulos	Intervalo(inteiro) [0-100000]	Número de módulos.
Procedimentos	Intervalo(inteiro) [0-1000000]	Número de procedimentos.
Loc	Intervalo(inteiro) [0-1000000]	Número de linhas de código.
Loc(executáveis)	Intervalo(inteiro) [0-1000000]	Número de linhas de código executáveis.
Classes	Intervalo(inteiro) [0-10000]	Número de classes.
Estados	Intervalo(inteiro) [0-10000]	Número de estados de um diagrama de estados.
Transições	Intervalo(inteiro) [0-10000]	Número de transições de um diagrama de estados.
Operações	Intervalo(inteiro) [0-10000]	Número de operações de um modelo de operações.
Módulos por módulos	Razão(real) [0-10000]	Razão entre módulos.
Proceds por proceds	Razão(real) [0-10000]	Razão entre procedimentos.
Lcom por loc(executáveis)	Razão(real) [0-10000]	Razão entre linhas de comentários e linhas de código executáveis.
Lcom	Intervalo(inteiro) [0-1000000]	Número de linhas de comentários.
Pessoas	Intervalo(inteiro) [0-1000]	Número de pessoas.
Nível de conhecimento	Ordinal [1-nenhum, 2-pouco, 3-médio, 4-bom, 5-ótimo]	Nível de conhecimento.
Telas	Intervalo(inteiro) [0-10000]	Número de telas de um sistema.
Número de opções	Intervalo(inteiro) [0-10000]	Número de opções de menu de um sistema.

Unidade	Tipo Escala /Valores Válidos	Descrição (Continuação)
Linguagem de programação	Nominal [basic, c, clipper, cobol, fortran, pascal, outras]	Linguagem de programação de um módulo do sistema.
Tempo(horas)	Intervalo(real) [0-100000]	Tempo em horas.
Tempo(meses)	Intervalo(real) [0-1000]	Tempo em meses.
Data	Data	Data de ocorrência.
Plataforma	Nominal [Pc/DOS, Pc/Windows 3.x, Pc/Windows 9.x-Me,NT-2000, Pc/Linux, outras]	Plataforma operacional de um sistema.
Executada	Nominal [Sim, Não]	Indica se uma determinada atividade foi executada
Sentenças	Intervalo(inteiro) [0-1000]	Sentenças do modelo de ciclo de vida

Para que um modelo de qualidade na abordagem SQUID seja considerado completo há a necessidade de se associar a cada atributo externo uma unidade de medida e uma regra de contagem (os atributos internos terão associados suas respectivas unidades na etapa de integração dos modelos de qualidade e de desenvolvimento).

A Tabela 5.11 descreve a associação de unidades de medidas e regras de contagem a cada atributo externo pertencente ao modelo de qualidade proposto.

Tabela 5.11 – Associação de Unidades de Medida aos Atributos Externos

Atributo Externo	Unidade	Regra de Contagem
Tempo na análise do código	Tempo (horas)	Tempo em horas na análise do código fonte do sistema (soma dos tempos na análise de cada módulo do sistema).
Tempo na definição dos temas	Tempo (horas)	Tempo em horas na definição dos temas do sistema.
Tempo na elaboração do modelo de ciclo de vida	Tempo (horas)	Tempo em horas na elaboração do modelo de ciclo de vida do sistema.
Tempo na elaboração do modelo de objetos	Tempo (horas)	Tempo em horas na elaboração do modelo de objetos do sistema (soma dos tempos na elaboração do modelo de objetos para cada tema definido para o sistema).
Tempo na elaboração do quadro de índice de procedimentos	Tempo (horas)	Tempo em horas na elaboração do quadro de índice de procedimentos do sistema.
Tempo na elaboração do quadro de operações-procedimentos de implementação	Tempo (horas)	Tempo de em horas na elaboração do quadro de operações-procedimentos de implementação (interface e temas do sistema).
Tempo na elaboração do quadro de chamadas de procedimentos	Tempo (horas)	Tempo em horas na elaboração do quadro de chamadas de procedimentos do sistema.
Tempo na coleta de informações	Tempo (horas)	Tempo em horas na obtenção de informações sobre o sistema (coleta de informações sobre o domínio da aplicação, entrevistas com usuários e desenvolvedores, etc).

Atributo Externo	Unidade	Regra de Contagem (Continuação)
Tempo no entendimento das funcionalidades do sistema	Tempo (horas)	Tempo em horas para entender as funcionalidades do sistema.
Tempo de projeto	Tempo (meses)	Tempo requerido no projeto de engenharia reversa.

Uma vez feita essa associação o modelo de qualidade SQUID está completo.

Na próxima seção descreve-se a última etapa de modelagem de um processo utilizando a abordagem SQUID: a integração dos modelos de qualidade e de desenvolvimento.

5.5 – Integração dos Modelos de Qualidade e de Desenvolvimento

A integração dos modelos de qualidade e de desenvolvimento (*Collate Model*) consiste em associar atributos internos de um modelo de qualidade aos objetos de projeto (de um modelo de desenvolvimento) juntamente com unidades de medidas e regras de contagem.

Pontos de Revisão (Integração)

A Tabela 5.12 mostra a associação dos Pontos de Revisão do modelo de desenvolvimento proposto com suas métricas. No caso de Pontos de Revisão, o único atributo interno associado é a data de sua ocorrência, indicando o final de uma determinada fase no processo de engenharia reversa.

Tabela 5.12 – Integração dos Modelos (Pontos de Revisão)

Pontos de Revisão	Atributo Interno	Unidade	Regra de Contagem
1. Recuperação de Informações	Data final da fase de recuperação de informações	Data	Data final da fase de recuperação de informações sobre o sistema.
2. Recuperação de visões funcionais	Data final da fase de recuperação de visões funcionais	Data	Data final da fase de recuperação de visões funcionais do sistema.
3. Recuperação de visões estruturais	Data final da fase de recuperação de visões estruturais	Data	Data final da fase de visões estruturais do sistema.
4. Gerenciamento do projeto	Data final do projeto	Data	Data final do projeto de engenharia reversa.

Atividades de Desenvolvimento (Integração)

A Tabela 5.13 mostra a associação das Atividades de Desenvolvimento com seus atributos internos. Também neste caso, há somente um atributo interno associado a cada atividade, indicando se uma determinada atividade já foi executada (uma atividade é considerada executada quando todos seus produtos foram gerados).

Tabela 5.13 – Integração dos Modelos (Atividades de Desenvolvimento)

Atividades	Atributo Interno	Unidade	Regra de Contagem
1.1 Obter informações existentes sobre o sistema	Atividade executada	Executada	Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados).
2.1 Elaborar modelo de ciclo de vida do sistema			
2.2 Elaborar modelo de operações do sistema			
2.3 Definir temas			
2.4 Elaborar modelo de objetos do sistema			
3.1 Elaborar quadro de chamadas de procedimentos	Atividade executada	Executada	Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados).
3.2 Elaborar quadro de índice dos procedimentos			
3.3 Elaborar quadro de operações – procedimentos de implementação			
4. Obter informações sobre o gerenciamento projeto			

Produtos (Integração)

A Tabela 5.14 mostra a associação dos produtos gerados pelas atividades de desenvolvimento às suas métricas internas identificadas.

Tabela 5.14 – Integração dos Modelos (Produtos)

Produtos	Atributo Interno	Unidade	Regra de Contagem
1.1 Informações sobre o sistema (executável)	Tipo de plataforma	Plataforma	Plataforma operacional na qual o sistema executa.
	Opções de menu	Número de opções	Número total de opções do menu do sistema.
	Telas do sistema	Telas	Número total de telas do sistema.
	Profundidade menu	Número de opções	Nível de profundidade do menu do sistema.

Produtos	Atributo Interno	Unidade	Rêgra de Contagem
1.2 Informações sobre o sistema (código fonte)	Densidade de comentários (sistema)	Lcom por loc (executáveis)	Razão entre linhas de comentários e linhas executáveis no código fonte do sistema.
	Densidade de módulos dependentes de hardware	Módulos por módulos	Razão entre número de módulos dependentes de hardware e número de módulos do sistema.
	Linhas de comentários (sistema)	Lcom	Soma das linhas de comentários dos módulos que compõem o sistema.
	Módulos do sistema	Módulos	Número total de módulos do sistema.
	Tamanho do sistema (loc)	Loc	Soma das linhas de código dos módulos que compõem o sistema.
	Tamanho do sistema (loc executáveis)	Loc (executáveis)	Soma das linhas de código executáveis dos módulos do sistema.
	Módulos dependentes de hardware	Módulos	Número de módulos que contém implementações dependentes do hardware.
1.3 Informações sobre o sistema (módulos)	Densidade de comentários (módulo)	Lcom por loc (executáveis)	Razão entre linhas de comentários e linhas executáveis no código fonte de um módulo.
	Linguagem de implementação	Linguagem de programação	Linguagem de programação utilizada na implementação de um módulo.
	Linhas de comentários (módulo)	Lcom	Linhas de comentários de um módulo.
	Fan-out	Módulos	Número de módulos que são chamados por um módulo.
	Fan-in	Módulos	Número de módulos chamando um módulo.
	Procedimentos (módulo)	Procedimentos	Número de procedimentos contidos em um módulo.
	Tamanho do módulo (loc)	Loc	Número de linhas de código do módulo.
	Tamanho do módulo (loc executáveis)	Loc (executáveis)	Número de linhas de código executáveis do módulo.
2.1 Modelo de ciclo de vida	Número de estados	Estados	Número de estados do diagrama de estados.
	Número de sentenças	Sentenças	Número de sentenças do modelo de ciclo de vida
	Número de transições	Transições	Número de transições do diagrama de estados.
2.2 Modelo de operações	Número de operações	Operações	Número de operações do modelo de operações.
2.3 Temas do sistema	Classes (tema)	Classes	Número de classes de um tema do sistema.

Produtos	Atributo Interno	Unidade	Regra de Contagem (cont.)
2.4 Modelo de objetos	Classes (modelo de objetos)	Classes	Soma do número de classes de cada tema definido para o sistema.
3.1 Quadro de chamadas de procedimentos	Fan-out (procedimentos)	Procedimentos	Número de procedimentos chamados por um procedimento.
	Fan-in (procedimentos)	Procedimentos	Número de procedimentos chamando por um procedimento.
3.2 Quadro de índice de procedimentos	Procedimentos (sistema)	Procedimentos	Número total de procedimentos contidos nos módulos do sistema.
3.3 Quadro de operações – procedimentos de implementação (interface)	Densidade de procedimentos (interface)	Proceds por proceds	Razão entre o número de procedimentos associados à interface e o número de procedimentos do sistema.
	Procedimentos (interface)	Procedimentos	Número de procedimentos identificados como implementação da interface do sistema.
3.4 Quadro de operações – procedimentos de implementação (temas)	Densidade de procedimentos (tema)	Proceds por proceds	Razão entre o número de procedimentos associados a um tema e o número de procedimentos do sistema.
	Procedimentos (tema)	Procedimentos	Número de procedimentos identificados como de implementação de um tema.
4. Informações sobre o gerenciamento do projeto	Data de início do projeto	Data	Data de início do projeto de engenharia reversa do sistema.
	Pessoas alocadas ao projeto	Pessoas	Número de pessoas responsáveis pela engenharia reversa do sistema.
	Nível de conhecimento	Nível de conhecimento	Nível de conhecimento do pessoal responsável pela engenharia reversa em relação à(s) linguagem(ns) de implementação dos módulos do sistema.

Todo este processo de modelagem descrito neste capítulo consiste da etapa de configuração. É a etapa mais crítica da abordagem SQUID porque envolve a modelagem de um modelo de desenvolvimento, a definição de um modelo de qualidade associado e a escolha de métricas adequadas, que serão usadas durante todo processo de especificação, planejamento, controle, avaliação da qualidade e também para a formação da base histórica.

5.6 – Considerações Finais

Neste capítulo foi feita a modelagem do método de engenharia reversa FUSION-RE/I segundo a abordagem SQUID, ou seja definido um modelo de desenvolvimento e proposto um modelo de qualidade associado, estabelecidas unidades de medidas e realizada a integração dos modelos de desenvolvimento e de qualidade.

No próximo capítulo será descrito como introduzir a modelagem proposta neste trabalho na ferramenta SQUID e diretrizes de como utilizá-la na em um projeto de engenharia reversa.

Capítulo 6 – Instanciando a Modelagem Proposta na Ferramenta SQUID

6.1 – Considerações Iniciais

Para a utilização da ferramenta SQUID em um projeto, é necessário primeiro a realização da modelagem de um modelo de desenvolvimento e ter associado a este um modelo de qualidade. Neste capítulo será ilustrado como introduzir a modelagem realizada no capítulo anterior para o método de engenharia reversa FUSION-RE/I e diretrizes de como instanciar um projeto de engenharia reversa utilizando essa modelagem e a ferramenta SQUID.

6.2 – Configurando a Ferramenta SQUID

Para utilizar a ferramenta SQUID em um projeto de engenharia reversa utilizando a modelagem proposta no capítulo anterior, é necessário primeiro introduzi-la na ferramenta na fase denominada *configuração da ferramenta* (Figura 6.1).

Figura 6.1 – Tela Inicial da Ferramenta SQUID (Configuração da Ferramenta)

A seguir são descritos os procedimentos necessários para a configuração da ferramenta:

- Instanciar o modelo de desenvolvimento (Figura 6.2) utilizando a Tabela 5.1, Tabela 5.2 e Tabela 5.3;

Figura 6.2 – Introduzindo o Modelo de Desenvolvimento

- Para a introdução de atributos de qualidade na ferramenta SQUID é necessário primeiro criar categorias de atributos externos e internos. Depois de criadas as categorias de atributos, pode-se então entrar com os atributos de acordo com sua categorização (Tabela 5.4 e Tabela 5.7). A Figura 6.3 ilustra essa atividade.

Figura 6.3 – Introduzindo Atributos de Qualidade

- Criar as unidades de medidas necessárias para quantificar os atributos de qualidade (Tabela 5.10). A Figura 6.4 mostra como é representada uma unidade de medida na ferramenta SQUID.

Figura 6.4 – Criando Unidades de Medida

- Na introdução do modelo de qualidade são necessários:
 - a) Introduzir as características e subcaracterísticas de qualidade externas e internas (Tabela 5.5 e Tabela 5.8);
 - b) Associar a cada características ou subcaracterísticas, seus respectivos atributos (Tabela 5.6 e Tabela 5.9);
 - c) Associar unidades de medida e regras de contagem aos atributos externos pertencentes ao modelo de qualidade (Tabela 5.11).

A Figura 6.5 mostra uma visão parcial do modelo de qualidade proposto (características, subcaracterísticas e atributos) quando introduzido na ferramenta SQUID.

Figura 6.5 -- Representado o Modelo de Qualidade na Ferramenta SQUID

- Integrar os modelos de qualidade e de desenvolvimento através da associação de atributos internos do modelo de qualidade aos objetos de projeto do modelo de desenvolvimento (Tabela 5.12, Tabela 5.13 e Tabela 5.14). Finalizada a associação, é necessário clicar no botão *complete* para indicar a ferramenta que a integração está completa (Figura 6.6). Após esta etapa ambos modelos (qualidade e desenvolvimento) estão integrados na ferramenta e não podem ser mais alterados. Uma vez feita a integração, pode-se associar projetos ao modelo de qualidade, e seus subsistemas (product portions) e subprojetos ao modelo de desenvolvimento integrado. Esta associação é necessária para assegurar que se tenha uma área na base de dados da ferramenta, para armazenar os valores dos dados de projetos e subsistemas [SQUID 98].

Figura 6.6 – Integração dos Modelos de Qualidade e de Desenvolvimento

Para utilizar a ferramenta em projetos, é necessário (ainda na fase de configuração da ferramenta) definir tipos de projetos e criar uma base de dados para cada novo projeto (pertencente a um tipo de projeto). Para cada projeto novo é então definido um nome, sua descrição, o modelo de qualidade associado e o tipo de projeto ao qual se refere (Figura 6.7).

Figura 6.7 – Criando um Novo Projeto

Criado a base de dados, o próximo passo é especificar os requisitos de qualidade externos para o projeto.

6.3 – Especificação da Qualidade

Quando se tem um novo projeto, a ferramenta SQUID espera que se estabeleçam os requisitos de qualidade do projeto em termos de valores objetivos para os atributos externos.

No processo de definição dos requisitos de qualidade, é necessário primeiro definir o produto em termos de subsistemas (*product portions*). Um sistema submetido à engenharia reversa pode ser composto por vários subsistemas (Figura 6.8). Neste contexto, subsistemas são partes de um produto de software que possuem diferentes requisitos de qualidade. Os subsistemas definidos deverão ser baseados sobre uma visão dos principais componentes de software do sistema.

Figura 6.8 – Instanciando Subsistemas (Product Portions)

Uma vez definidos os subsistemas, é necessário associar cada requisito de qualidade do projeto ao subsistema a qual se aplica.

Finalizado este procedimento, a ferramenta SQUID permite associar cada requisito com uma característica de qualidade do modelo de qualidade usado. Estas atividades dependem fortemente da experiência pessoal. Contudo, após terminado o mapeamento dos requisitos para os subsistemas, a ferramenta SQUID ajudará a associar cada requisito com uma característica ou subcaracterística de qualidade, de acordo com o modelo de qualidade. Uma vez feita a união entre (sub)característica de qualidade e um requisito, a ferramenta SQUID irá identificar um ou mais atributos externos (pertencentes ao modelo de qualidade associado) sobre o qual pode-se definir valores objetivos quantificáveis (Figura 6.9).

Figura 6.9 – Definindo Requisitos de Qualidade

6.4 – Avaliando a Viabilidade dos Requisitos de Qualidade

A avaliação da viabilidade dos requisitos de qualidade baseia-se na comparação da qualidade alcançada no passado, de subsistemas de produtos similares, com os requisitos de qualidade do subsistema atual. Por padrão, a ferramenta SQUID assume que subsistemas associados com o mesmo modelo de qualidade são “similares” [SQUID 98]. Assim, SQUID construirá uma base de estimativa de subsistemas que tem o mesmo modelo de qualidade. Para este propósito se utiliza um *Índice de Similaridade* definido pela ferramenta. Como, por não contar com subsistemas de projetos anteriores e pelas limitações da versão utilizada neste trabalho, não foi possível aprofundar o estudo sobre este recurso.

6.5 – Planejamento da Qualidade

Os atributos de qualidade internos são propriedades mensuráveis de um produto de software ou do processo de desenvolvimento que influenciam a qualidade do produto final [NBR 13596]. Durante o planejamento da qualidade é necessário estabelecer valores objetivos para os atributos internos com a finalidade de monitorar o progresso dos mesmos. Para isso é necessário primeiramente definir a visão interna de cada subsistema.

SQUID assume que cada subsistema definido durante a etapa de especificação da qualidade será desenvolvido por um subprojeto específico. Quando se define um subprojeto, se deve relacionar um subsistema do produto em particular a um modelo de desenvolvimento específico, o qual deve ter sido integrado com o modelo de qualidade durante o processo de configuração da ferramenta.

No caso da aplicação de um projeto de engenharia reversa utilizando os modelos definidos neste trabalho, deve-se relacionar cada subsistema (estabelecido na etapa de planejamento) a um subprojeto que utilize o modelo de desenvolvimento do processo de engenharia reversa (descrito na Seção 5.2 do Capítulo 5), criando assim uma visão interna de cada subsistema (Figura 6.10).

Figura 6.10 – Definindo a Visão Interna de um Subsistema

A ferramenta SQUID considera o planejamento da qualidade como a definição de valores objetivos quantificáveis para os atributos internos associados aos produtos intermediários e atividades de desenvolvimento, junto aos quais o progresso será monitorado [SQUID 98].

Para cada subsistema deve-se instanciar os objetos de projeto (pertencentes ao modelo de desenvolvimento) nos quais deseja-se medir a qualidade. Cada objeto de projeto que foi definido como não repetível no modelo de desenvolvimento poderá ter apenas uma instância enquanto os que foram definidos como repetíveis poderão ter várias instâncias do mesmo objeto de projeto. A Figura 6.11 ilustra a instanciação de vários objetos de projeto *Informações sobre o sistema*

(módulos) onde cada instância representa um módulo (código fonte) pertencente ao subsistema definido.

Figura 6.11 – Criando Instâncias de Objetos de Projeto para um Determinado Subsistema

Depois de instanciados os objetos de projeto pertencentes a cada subsistema, pode-se definir quais são os valores objetivos para seus atributos internos (Figura 6.12).

Figura 6.12 – Atribuindo Valores Objetivos aos Atributos Internos

Num processo de engenharia reversa, os valores objetivos em relação ao sistema executável e aos módulos do sistema podem representar a qualidade desejável de um software (em relação a um padrão de qualidade) e os valores obtidos indicam o quanto o software se aproxima (ou se distância) dessa qualidade. É necessário então, primeiro, haver um estudo de quais valores objetivos devem ser definidos para cada atributo de um projeto particular.

Através da coleta de dados de vários projetos de engenharia reversa que utilizem o mesmo modelo de qualidade e de desenvolvimento, pode-se conduzir um estudo para tentar definir um conjunto de valores objetivos para os atributos internos e determinar categorias levando em consideração as características de cada projeto. A própria ferramenta SQUID fornece muitos recursos estatísticos que poderiam ajudar nesse estudo (Apêndice E).

6.6 – Medidas Atuais

Uma vez planejado a qualidade do projeto, este pode ser monitorado com respeito aos valores objetivos estabelecidos na etapa de planejamento da qualidade. Para isto é necessário primeiro introduzir os valores correntes obtidos para cada atributo.

Para se introduzir os valores atuais de atributos externos de um projeto, é necessário primeiro selecionar de qual subsistema a medida pertence. A Figura 6.13 ilustra como os valores atuais para os atributos externos são introduzidos na ferramenta SQUID.

Figura 6.13 – Atribuindo Valores Atuais aos Atributos Externos

Para ingressar os valores atuais aos atributos internos de um projeto, primeiramente é necessário selecionar o subsistema, o objeto de projeto, a instância do objeto de projeto e por último o atributo de qualidade ao qual a medida pertence. A Figura 6.14 mostra como as medidas atuais de atributos internos de um projeto são introduzidas na ferramenta SQUID.

Figura 6.14 – Atribuindo Valores Atuais aos Atributos Internos

Após introduzidos os valores atuais de atributos de qualidade, pode-se iniciar o próximo passo do processo de qualidade SQUID, o controle da qualidade.

6.7 – Controle da Qualidade

A ferramenta SQUID considera o monitoramento da qualidade como um meio de avaliar o desenvolvimento de um produto através dos valores objetivos definidos durante o planejamento da qualidade e decidindo reações apropriadas para quaisquer desvios [SQUID 98].

A ferramenta SQUID utiliza duas estratégias para o monitoramento da qualidade de um projeto, que são:

- Identificar componentes com valores de atributos não usuais;
- Monitorar os valores atuais com relação aos valores objetivos planejados quantitativamente.

Através de métodos estatísticos a ferramenta é capaz de identificar componentes que fogem da tendência geral da amostra analisada. A Figura 6.15 mostra um exemplo onde cada módulo de um subsistema submetido à engenharia reversa é analisado com relação ao número de linhas de

código e ao número comentários que cada módulo possui. Os pontos com um círculo indicam os módulos que fogem da tendência geral dos demais módulos.

Figura 6.15 – Detecção de Componentes Anômalos

Para o monitoramento da qualidade, a ferramenta SQUID disponibiliza vários relatórios com os valores objetivos, estimados e valores atuais, e por outro lado provê recursos estatísticas que permitem realizar análise de regressão de uma, duas ou várias variáveis. Esta é a etapa onde se pode obter o maior proveito da ferramenta SQUID, ajudando assim a controlar e avaliar o desenvolvimento de produtos de software. O Apêndice E mostra algumas telas geradas pela ferramenta ilustrando esses recursos.

6.8 – Avaliação da Qualidade

Para a realização deste trabalho de mestrado foi utilizada a versão 1.22a da ferramenta SQUID, a qual não implementava ainda todas as funcionalidades, incluindo a avaliação da qualidade (Figura 6.16), e por este motivo não foi possível fazer um estudo mais detalhado de como a ferramenta auxilia nesta etapa final do processo de qualidade. Neste caso, uma alternativa seria criar uma tabela com todos os valores objetivos definidos na etapa de planejamento (para os atributos de qualidade externos e internos), e compará-los com todos os valores correntes associados aos atributos de qualidade externos e internos obtidos na etapa final do processo de engenharia reversa. Através da análise dessas informações ao decorrer de vários projetos, poderia ser feito um estudo de quais valores objetivos deveriam ser definidos dependendo do projeto e da capacidade da empresa. Esses dados também poderiam indicar a necessidade de

adequação de alguns processos da empresa, implicando numa análise de áreas problemáticas e alternativas de soluções (melhoria do processo).

Figura 6.16 – Avaliação da Qualidade (Não Implementado)

6.9 – Considerações Finais

Neste capítulo foram descritos como instanciar a modelagem realizada neste trabalho e diretrizes de como aplicá-la em um projeto utilizando a ferramenta SQUID. Foram abordadas as fases de configuração da ferramenta, especificação da qualidade, planejamento da qualidade, controle da qualidade e avaliação da qualidade, focalizando no processo de engenharia reversa modelado neste trabalho.

No capítulo a seguir são apresentadas as conclusões deste trabalho, bem como futuros desenvolvimentos que podem advir do mesmo.

Capítulo 7 – Conclusões e Trabalhos Futuros

7.1 – Conclusões

A avaliação de produtos de software constitui uma área em que a demanda cresceu significativamente, paralelamente ao estabelecimento de normas, padrões e ao desenvolvimento de métodos e técnicas para auxiliar no processo de avaliação. Os métodos de avaliação de qualidade de produto de software contribuem na obtenção de produtos de software melhores e mais confiáveis.

Através de pesquisas realizadas recentemente [MCT 00], observou-se que 63% das empresas de desenvolvimento de software no Brasil são micro ou pequenas empresas e que ainda não possuem uma cultura voltada para a questão da qualidade.

Devido à emergente preocupação que se tem sentido no mercado, em busca da qualidade, muitas empresas que desenvolvem software têm procurado melhorar seu processo de desenvolvimento e a qualidade de seus produtos (existentes ou em desenvolvimento).

A manutenção de software é um dos problemas enfrentado por muitos profissionais de software, principalmente naqueles softwares chamados de “legados”, os quais geralmente têm uma grande importância dentro de uma organização e que foram desenvolvidos há vários anos possuindo pouca documentação disponível. Com o desenvolvimento de novas arquiteturas e melhorias nos métodos e linguagens de programação, incluindo programação orientada a objetos, existe uma forte motivação para a engenharia reversa e a reengenharia de sistemas existentes com o objetivo de melhorar sua qualidade, preservar ou modificar sua funcionalidade, enquanto utiliza as últimas tecnologias.

Este trabalho procurou contribuir nesta perspectiva utilizando uma nova abordagem para avaliação da qualidade, a abordagem SQUID, a qual está baseada nos conceitos e definições das normas NBR 13596, ISO 9001, ISO/IEC 14598 e no modelo CMM, e que envolve tanto a visão de qualidade de um produto como também do processo de desenvolvimento desse produto. Essa abordagem foi utilizada na modelagem de um processo de engenharia reversa, permitindo assim seu gerenciamento e controle, e visando à obter indicativos da qualidade atual do produto e dos artefatos produzidos por esse processo. Na modelagem foi utilizado o método de engenharia reversa orientado a objetos FUSION-RE/I que tem como objetivo auxiliar na atividade de manutenção, recuperando visões funcionais e estruturais de um sistema. Com uma base histórica de projetos realizados, uma empresa poderá levantar suas reais capacidades de desenvolvimento de projetos e/ou possíveis áreas problemáticas e buscar alternativas, auxiliando assim numa

melhora de seu processo. Outro objetivo implícito neste trabalho foi através da utilização da abordagem SQUID, incentivar a adoção da prática de coleta de métricas, mesmos que as mais simples, com a intenção de introduzir essa prática nas micro e pequenas empresas de software nacionais as quais não possuem essa cultura.

Neste trabalho foram realizados:

- Um estudo dos modelos de avaliação da qualidade de produtos de software e de modelos de avaliação e melhoria da qualidade de processos de software;
- Um estudo sobre engenharia reversa e o método de engenharia reversa orientado a objetos FUSION-RE/I;
- A engenharia reversa parcial de um sistema, a qual deu subsídios para um melhor entendimento do processo de engenharia reversa e da aplicação do método FUSION-RE-I;
- Um estudo da abordagem SQUID (método e como a ferramenta apóia este método). Durante o desenvolvimento deste trabalho, foi realizado um curso introdutório sobre a abordagem SQUID, ministrado por dois pesquisadores membros projeto SQUAD (Stefano de Panfilis – Engineering Ingegneria Informatica / Itália e Jørgen Bøegh – DELTA Danish Electronics, Ligth & Acoustics / Dinamarca), como parte das atividades do *Second SQUAD Week*, ocorrido no CTI (Fundação Centro Tecnológico para Informática) de Campinas. Também como atividade deste trabalho de mestrado, foi realizada uma apresentação da ferramenta SQUID para o programa de Pós-Graduação em Informática Aplicada da PUC/PR (Pontificia Universidade Católica do Paraná), Campus Curitiba;
- A modelagem de um processo de engenharia reversa utilizando o método FUSION-RE/I segundo a abordagem SQUID, que envolveu a modelagem de um modelo de desenvolvimento (identificando pontos de revisão, atividades de desenvolvimento e produtos gerados por essas atividades), a definição um modelo de qualidade (características e subcaracterísticas de qualidade externas e internas, atributos de qualidade externos e internos, unidades de medida) e a integração desses modelos;
- Uma descrição de como instanciar a modelagem SQUID realizada para o método FUSION-RE/I e diretrizes de como utilizá-la em um projeto de engenharia reversa utilizando a ferramenta SQUID.

Algumas características da abordagem SQUID não puderam ser estudadas com mais detalhes, devido ao fato da versão da ferramenta que apóia o método, utilizada para a realização deste trabalho, implementar parcialmente as funcionalidades que o método SQUID propõe.

Por ser uma abordagem recente, há poucas publicações existentes sobre o assunto. Vários estudos de sua aplicação e validação ainda estão em andamento, como o projeto VALSE (VALidating Squid in real Environments) [VALSE] da comunidade européia.

7.2 – Trabalhos Futuros

Algumas linhas de pesquisa que podem dar continuidade a este trabalho são:

- Aplicar a modelagem proposta em alguns projetos. Através da análise dos resultados desses projetos, realizar um estudo de como a abordagem SQUID e a aplicação da modelagem proposta nesse trabalho contribuem para a avaliação da capacidade da empresa e na obtenção de eventuais áreas problemáticas no processo. Para se utilizar os recursos estatísticos e de estimativas que a ferramenta fornece, de forma que os dados coletados tenham algum valor significativo e possam servir de referência para uma análise estatística e estimativas, a abordagem SQUID sugere ao menos 7 projetos anteriores utilizando o mesmo modelo de qualidade e de desenvolvimento;
- Incorporar novas métricas ao modelo de qualidade e modelar um processo de reengenharia utilizando a abordagem SQUID;
- Descrever um modelo de maturidade de capacidade para empresas que realizam engenharia reversa de sistemas (RE-CMM – Reverse Engineering-Capability Maturity Model) e suas áreas-chave de processo para cada nível de maturidade. Juntamente pode-se fazer um estudo de como a utilização da abordagem SQUID e de um processo de engenharia reversa como o descrito neste trabalho podem auxiliar essas empresas a elevarem seu nível de maturidade.
- Caracterizar um processo de desenvolvimento de software para pequenas empresas no contexto brasileiro e modelá-lo segundo a abordagem SQUID, ou seja definir um modelo de desenvolvimento e um modelo de qualidade adequados à realidade da maioria das empresas de desenvolvimento de software nacionais e avaliar a adequação do uso da ferramenta SQUID;

Referências Bibliográficas

- [Andrade 96] Andrade, A.L.P. et al., "Aplicação da Norma ISO/IEC 12119 na Avaliação da Qualidade de Produtos de Software", *VII Conferência Internacional de Tecnologia de Software (VII CITS)*, Curitiba, Junho, 1996.
- [Atkinson 97] Atkinson, Colin, "Adapting the Fusion Process: How Fusion can help users in creating UML diagrams for software development", *Object Development*, November, 1997.
- [Basili 84] Basili, V.R.; Weiss, D.M., "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions on Software Engineering*, v. SE-10, n. 6, November, 1984, pp. 728-738.
- [Basili 86] Basili, V.R.; Selby, R.W.; Hutchens, D.H., "Experimentation in Software Engineering", *IEEE Transaction on Software Engineering*, v. 12, n. 7, 1986, p. 733-743.
- [Basili 88] Basili, V.R.; Rombach, H.D., "The Tame Project: Towards Improvement-Oriented Software Environments", *IEEE Transactions on Software Engineering*, v. 14, n. 6, 1988, p. 758-773.
- [Basili 90] Basili, V.R., "Viewing Maintenance as Reuse-Oriented Software Development", *IEEE Software*, v. 7, n.1, January, 1990, p. 19-25.
- [Basili 91] Basili, V.; Musa, J., "The Future Engineering of Software: A Management Perspective", *IEEE Computer*, 1991.
- [Basili 95] Basili, V.R.; Caldiera, G.; Rombach, H.D., "Goal Question Metric Paradigm", *Encyclopedia of Software Engineering*, Jonh Wiley & Sons, New York, v. 1, 1995, p. 527-532.
- [Belchior 92] Belchior, A.D.; Rocha, A.R.C., *Características de Qualidade de Programas*, Relatório Técnico ES-265/92, COPPE/UFRJ, Julho, 1992.
- [Benedusi 92] Benedusi, P.; Cimitile, A.; Carlini, U., "Reverse Engineering Processes, Design Document Production, and Structure Charts", *Journal of Systems and Software*, v. 19, 1992, p. 225-245.

- [Bennett 98] Bennett, K.H.; Munro, M., "DERE: Evaluating Reverse Engineering", Computer Science Technical Report CS/TR/1998/5, Computer Science Department, University of Durham, Url: <http://www.dur.ac.uk/CompSci/technical-reports.html> [Acesso em Nov/2000].
- [Biggerstaff 89] Biggerstaff, T.J., "Design Recovery for Maintenance and Reuse", *IEEE Computer*, v. 22, n. 7, 1989, p.36-49.
- [Bøegh 99] Bøegh, J. et al., "A Method for Software Quality Planning, Control, and Evaluation", *IEEE Software*, March/April, 1999.
- [Booch 00] Booch, G.; Rumbaugh, J.; Jacobson, I, *UML – Guia do Usuário*, Editora Campus, 2000.
- [Briand 96] Briand, L., "Goal-Driven Definition of Products Metrics Based on Properties", <http://www.iese.fhg.de/Publications/TechRep15-CS-TR-3346.html> , Modificado em 26/07/96 [Acesso em Jan/1999].
- [Chikofsky 90] Chikofsky, E.J.; Cross II, J.H., "Reverse Engineering and Design Recovery: A Taxonomy", *IEEE Software*, v. 7, n. 1, 1990, p.13-17.
- [Choi 90] Choi, S.C.; Scacchi, W., "Extracting and Reestructuring the Design of Large Systems", *IEEE Software*, v. 7, n. 1, 1990, p. 66-71.
- [Cimitile 99] Cimitile, Aniello et al., "Identifying Objects in Legacy Systems using Design Metrics", *The Journal of Systems and Software*, v. 44, n. 3, January ,1999.
- [CMM] SEI/CMU, "Capability Maturity Model for Software", <http://www.sei.cmu.edu/cmm/cmm.html> [Acesso em Jan/1999].
- [CMMI] SEI/CMU, "Capability Maturity Model Integration", <http://www.sei.cmu.edu/cmmi/products/models.html> [Acesso em Abr/2001].
- [Coallier 94] Coallier, F., "How ISO 9001 Fits into the Software World", *IEEE Software*, January, 1994.
- [Colanzi 99] Colanzi, T.E., *Uma Abordagem Integrada de Desenvolvimento e Teste de Software Baseada na UML*, Dissertação de Mestrado, ICMC-USP, 1999.

- [Coleman 96] Coleman, D. et al., *Desenvolvimento Orientado a Objetos: O Método Fusion*, Editora Campus, 1996.
- [Costa 97] Costa, R.M., *FUSION-REI/I: Um Método de Engenharia Reversa para Auxiliar a Manutenção de Software*, Dissertação de Mestrado, ICMC-USP, 1997.
- [Crosby 79] Crosby, P.B., *Quality is Free: The Art of Making Quality Certain*, New York, McGraw Hill, 1979.
- [Curtis 92] Curtis, B., "Maintaning the Software Process", *IEEE Conference on Software Maintenance*, Orlando, 1992, p. 2-8.
- [FAMOOS 01] FAMOOS, "Framework-based Approach for Mastering Objectct-Oriented Software Evolution", url: <http://www.iam.unibe.ch/~famous> [Acesso em Fev/2001].
- [Feltrim 99a] Feltrim, V.D., *Apoio à Documentação de Engenharia Reversa de Software por meio de Hipertextos*, Dissertação de Mestrado, ICMC-USP, 1999.
- [Feltrim 99b] Feltrim, V.D., *Documentação do sistema SASHE recuperado por meio de engenharia reversa*, Relatório Técnico n. 90, ICMC-USP, 1999.
- [Fenton 97] Fenton, Norman E.; Pfleeger, Shari L., *Software Metrics: A Rigorous and Practical Approach*, Second Edition, International Thomson Computer Press, 1997.
- [Fiorini 98] Fiorini, S.T.; Von Staa, A.; Baptista, R.M., *Engenharia de Software com CMM*, Brasport, 1998.
- [Gall 94] Gall, Harald C. et al., "Balancing in Reverse Engineering and in Object-Oriented Systems Engineering to Improve Reusability and Maintainability", *18th IEEE Computer Software and Applications Conference (COMPSAC '94)*, 1994.
- [Gall 95] Gall, Harald C. et al, "Finding Objects in Procedural Programs: An Alternative Approach", In: *Working Conference on Reverse Engineering*, 2, Toronto-Canada, Proceedings, IEEE Computer Society Press, p. 208-216, 1995.

- [Gresse 96] Gresse, C.; Rombach, H.D.; Ruhe, G., "A Practical Approach for Building GQM-Based Measurement Programs: Lessons Learned from Three Industrial Case Studies", *X Simpósio Brasileiro de Engenharia de Software*, Tutorial, São Carlos, Outubro, 1996.
- [Harandi 90] Harandi, M.T.; Ning, J.Q., "Knowledge-Based Program Analysis", *IEEE Software*, v. 7, n. 1, 1990, p. 74-81.
- [Humphrey 89] Humphrey, W.S., *Managing the Software Process*, 1ª. ed., Massachusetts, Addison-Wesley, 1989.
- [ISO 9000-3] ISO/IEC, "ISO/IEC 9000-3 – Quality Management and Quality Assurance Standards, Part 3: Guidelines for the Application of ISO 9001 to Development, Supply and Maintenance of Software", 1991.
- [ISO 9001] ISO 9001 ABNT, "Modelo para Garantia de Qualidade em Projetos, Desenvolvimento, Produção, Instalação e Assistência Técnica", 1990.
- [ISO 9001a] ISO 9001:2000, "Quality Management Systems. Requirements", 2000.
- [ISO/IEC 12119] ISO/IEC, "ISO/IEC 12119, International Standard. Information Technology Software Packages – Quality Requirements and Testing", October, 1994.
- [ISO/IEC 14598-5] ISO/IEC, "ISO/IEC 14598-5, International Standard. Information Technology – Software Product Evaluation – Part 5: Process for Evaluators", May, 1996.
- [ISO/IEC 9126] ISO/IEC, "ISO/IEC 9126, International Standard. Information Technology – Software Product Evaluation, Quality Characteristics and Guidelines for their Use", 1991.
- [ISO/IEC 9126-3] ISO/IEC, "WD 9126-3, Information Technology – Software Quality Characteristics and Metrics – Part 3: Internal Metrics", 1997.
- [ISO/IEC TR 15504] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report)", <http://www.sqi.gu.edu.au/spice/> [Acesso em Jan/1999].
- [ISO/IEC TR 15504a] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 1: Concepts and introductory guide", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].

- [ISO/IEC TR 15504b] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 2: A model for process management", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504c] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 3: Rating process", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504d] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 4: Guide to conducting assessments", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504e] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 5: Construction, selection and use of assessments instruments and tools", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504f] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 6: Qualification and training of assessors", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504g] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 7: Guide for use in process improvement", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504h] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 8: Guide for use in determining supplier process capability", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [ISO/IEC TR 15504i] ISO/IEC, "ISO/IEC TR 15504: Information Technology – Software Process Assessment (Technical Report) – Part 9: Vocabulary", <http://www.sqi.gu.edu.au/spice> [Acesso em Jan/1999].
- [Jacobson 91] Jacobson, I.; Lingstrom, F., "Re-engineering of Old Systems to na Object-Oriented Architeture", *SIGPLAN Notices*, v. 26, n. 11, nov., 1991, p. 340-350.
- [Jones 97] Jones C., *Applied Software Measurement, Assuring Productivity and Quality*, 2ed., Prentice Hall, 1997.

- [Jubileu 99] Jubileu, A.P., *Aquisição de Conhecimento como Apoio ao Método de Engenharia Reversa FUSION-RE/I*, Dissertação de Mestrado, ICMC-USP, 1999.
- [Juran 70] Juran, J.M.; Gryna Jr., F.M., *Quality Planning and Analysis: From Product Development Through Use*, New York, McGraw Hill, 1970.
- [Kan 95] Kan, S.H., *Metrics and Models in Software Quality Engineering*, Massachusetts, Addison-Wesley, 1995.
- [Kirner 95] Kirner, T.G.; Rombach, H.D., "InTec-Ser: Innovative Technologies for Software Requirements Engineering", *Proceeding of the 3rd. German-Brazilian, Workshop on Information Technology*, Berlim, Germany, December 1995, p. 56-61.
- [Kirner 97] Kirner, T.G.; Abib, J.C., "Inspection of Software Requirements Documents: A Pilot Study", *Proceeding of the XV ACM International Conference on System Documentation – SIGDOC'97*, Salt Lake City, UT, USA, October 1997, p. 161-171.
- [Kitchenham 95] Kitchenham, B.A.; Pfleeger, S.L.; Fentin, N.E., "Towards a framework for software measurement validation", *IEEE Transaction on Software Engineering*, v. 21, n.12, December, 1995.
- [Klösch 96] Klösch, René R., "Reverse Engineering: Why and How to Reverse Engineer Software", *Proceedings of the California Software Symposium (CSS '96)*, University of Southern California, April, 1996, p. 92-99.
- [Lamprecht 94] Lamprecht, J.L., *ISO 9000 e o Setor de Serviços: Uma Interpretação Crítica das Revisões 1994*, Rio de Janeiro, Qualitymark, 1994.
- [Losavio 99] Losavio, F.; Chirinos, L., "Evaluación de la Calidad en el Desarrollo de Sistemas Interactivos", *X CITS*, PUC-PR, 1999.
- [Marciniat 94] Marciniat, J.J., *Encyclopedia of Software Engineering*, Wiley Interscience Publications, v. 2, 1994, p. 851-869.
- [McCall 77] McCall, J. et al., *Factors in Software Quality*, NTIS AD-A049-014,015,055, November, 1977.
- [MCT 00] Ministério da Ciência e Tecnologia, Secretaria de Política de Informática e Automação, "Qualidade e Produtividade no Setor de Software Brasileiro 1999", 2000.

- [NBR 13596] NBR 13596 ABNT, "Tecnologia de Informação – Avaliação de Produto de Software – Características de Qualidade e Diretrizes para o seu Uso", 1996.
- [NBR ISO/IEC 12119] NBR 12119 ABNT, "Tecnologia de Informação – Pacotes de Software – Teste e Requisitos de Qualidade", 1996.
- [NBR ISO/IEC 12207] NBR ISO/IEC 12207 ABNT, "Tecnologia da Informação – Processos de Ciclo de Vida de Software", 1999.
- [Oliveira 95] Oliveira, K.M., *Avaliação de Sistemas Especialistas*, Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro, 1995.
- [Oxford 86] *Dictionary of Computing*, Oxford University Press, 1986.
- [Paulk 93] Paulk, M.C. et al., *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, Carnegie Mellon University, CMU/SEI-93-TR-24, ESC-TR-93-177, Fevereiro, 1993.
- [Penteado 96] Penteado, R.D., *Um Método para Engenharia Reversa Orientada a Objetos*, Tese de Doutorado, IFSC-USP, 1996.
- [Pressman 00] Pressman, R. S., *Software Engineering – A Practitioner's Approach*, 5ª. ed., McGraw-Hill, 2000.
- [Quinaia 98] Quinaia, M.A., *Diretrizes para Reengenharia de Software com Características de Software Legado*, Dissertação de Mestrado, ICMC-USP, 1998.
- [Rocha 01] Rocha, A.R.C.; Maldonado, J.C.; Weber, K.C., *Qualidade de Software – Teoria e Prática*, Prentice Hall, 2001.
- [Rombach 87a] Rombach, H.D.; Basili, V.R., "Quantitative Assessment of Software Maintenance: Na Industrial Case Study", *Conference Software Maintenance*, Austin, TX, 1987, p. 134-144.
- [Rombach 87b] Rombach, H.D., "A Controlled Experiment on the Impact of Software Structure on Maintainability", *IEEE Transactions on Software Engineering*, v. SE-13, n. 3, 1987.
- [Rombach 89] Rombach, H.D., "Improving Software Maintenance Through Measurement", *Proceedings of the IEEE*, v. 77, n. 4, 1989, p.581-595.
- [Rombach 90] Rombach, H.D., "Design Measurement: Some Lessons Learned.", *IEEE Software*, v. 7, n. 2, March, 1990, p. 17-24.

- [Rosa 97] Rosa, P.G., *Modelo de Avaliação de Processo de Software para a Pequena Empresa*, Dissertação de Mestrado, ICMC-USP, 1997.
- [Rosenberg 00] Rosenberg, L.H., "Software Re-engineering", Software Assurance Technology Center, NASA, 2000, url: <http://satc.gsfc.nasa.gov/support/reengrpt.PDF> [Acesso em Ago/2000].
- [Rothery 93] Rothery, B., *ISO 9000*, Rio de Janeiro, Makron Books, 1993.
- [Sanches 93] Sanches, R., *A Influência do Software e de seu Processo de Manutenção no Esforço de Manutenção*, Tese de Doutorado, FEA/USP, 1993.
- [Sedall 99] Sendall, Shane; Strohmeier, Alfred, "UML-based Fusion Analysis", <http://glwww.epfl.ch/~sendall/papers/UML-based Fusion Analysis Aug 99.html> [Acesso em Nov/2000].
- [Shepperd 90] Shepperd, M., "Design Metrics: Na Empirical Analysis", *Software Engineering Journal*, v. 5, n. 1, 1990, p. 3-10.
- [Shepperd 92] Shepperd, M., "Products, Process and Metrics", *Information and Software Technology*, v. 34, n. 10, October, 1992.
- [Sneed 95] Sneed, H. M., "Planning the Reengineering of Legacy Systems", *IEEE Software*, v. 12, Janeiro, 1995, p. 24-34.
- [SQUID 98] Kitchenham, B.; Linkman, S.; Bøegh, J.; Pasquini, A., "SQUID Conceptual Handbook", *Second SQUAD Week*, CTI, Campinas, Outubro, 1998.
- [Systa 00] Systa, T., "Static and Dinamic Reverse Engineering Techniques for Java Software Systems", University of Tampere, 2000, url: <http://acta.uta.fi/pdf/951-44-4811-1.pdf> [Acesso em Set/2000].
- [TeamFusion 97] TeamFusion, Engineering Process Summary – HP's Latest Improvements to Fusion, HP, July, 1997.
- [Tilley 98] Tilley, Scott, *A Reverse-Engineering Environment Framework*, Technical Report CMU/SEI-98-TR-005, Software Engineering Institute, Carnegie Mellon University, 1998 [Obtido via Internet].
- [Tsukumo 96] Tsukumo, A.N., et al., "Modelos de Processo de Software: Visão Global e Análise Comparativa", *VII Confederação Internacional de Tecnologia de Software (VII CITS)*, Curitiba, Junho, 1996.

[Tsukumo 97]	Tsukumo, A.N., et al., "Qualidade de Software: Visões de Produto e Processo de Software", <i>VIII Confederação Internacional de Tecnologia de Software (VIII CITS)</i> , Curitiba, Junho, 1997.
[Valse]	Valse, "Validating SQUID in Real Environments", url: http://valse.eng.it [Acesso em Jan/2001].
[Weber 01]	Weber, K.C. et al., <i>Qualidade e Produtividade em Software</i> , 4ª. Ed., Makron Books, 2001.
[Weber 95]	Weber, K.C; Pinheiro, M., "Software Quality in Brasil", <i>Quality World Magazine</i> , v. 21, n. 11, The Institute of Quality Assurance (IQA), London, UK, November, 1995.
[Weber 97]	Weber, K.C. et al., <i>Qualidade e Produtividade em Software</i> , 2ª. ed., Makron Books, 1997.

Apêndices

Apêndice A – Relatório SQUID do Modelo de Desenvolvimento Proposto

Apêndice B – Relatório SQUID do Modelo de Qualidade Proposto

Apêndice C – Relatório SQUID da Integração dos Modelos de Qualidade e de Desenvolvimento

Apêndice D – Relatório SQUID das Unidades de Medidas Utilizadas

Apêndice E – Telas de Recursos Estatísticos Disponíveis na Ferramenta SQUID

Apêndice A – Relatório SQUID do Modelo de Desenvolvimento Proposto

Development Model Report

Report Type: Full structure report Report date: 10/06/01

Model: MD ER

Completed on: 10/06/01

Modelo de Desenvolvimento do Metodo de Engenharia Revesa FUSION-RE/I

Review Points

Recuperacao de informacoes - NOT repeating

Final da fase de recuperacao de informacoes existentes sobre o sistema.

Recuperacao de visoes funcionais - NOT repeating

Final da fase de recuperacao de visoes funcionais do sistema

Recuperacao de visoes estruturais - NOT repeating

Final da fase de recuperacao de visoes estruturais do sistema

Gerenciamento do projeto - NOT repeating

Final da fase de gerenciamento do projeto de engenharia reversa.

Activities

Obter informacoes existentes sobre o sistema - NOT repeating

Obter informacoes disponiveis do sistema

Elaborar modelo de ciclo de vida do sistema - NOT repeating

Elaborar o modelo de ciclo de vida do sistema a partir da interface do sistema

Elaborar modelo de operacoes do sistema - NOT repeating

Definir temas - NOT repeating

Elaborar modelo de objetos do sistema - NOT repeating

Elaborar quadro de chamadas de procedimentos - NOT repeating

Elaborar quadro de indice dos procedimentos - NOT repeating

Elaborar quadro op-proceeds de implementacao - NOT repeating

Obter informacoes sobre gerenciamento do projeto - NOT repeating

Deliverables

Informacoes sobre o sistema - executavel - NOT repeating

Informacoes obtidas sobre o sistema executavel

Informacoes sobre o sistema - codigo fonte - NOT repeating

Informacoes sobre o sistema - modulos - repeating

Informacoes sobre os modulos que compoem o sistema

Modelo de ciclo de vida - NOT repeating

Modelo de operacoes - NOT repeating

Temas do sistema - repeating

Quadro de chamadas de procedimentos - NOT repeating

Quadro de indice de procedimentos - NOT repeating

Quadro op-proceeds implementacao - interface - NOT repeating

Quadro op-proceeds implementacao - temas - repeating

Informacoes sobre o gerenciamento do projeto - NOT repeating

Modelo de objetos - NOT repeating

Apêndice B – Relatório SQUID do Modelo de Qualidade Proposto

Quality Model Report

Report Type: Full structure report Report date: 10/06/01

Quality Model: MQ ER

Completed on: 10/06/01

Modelo de Qualidade - Engenharia Reversa

Manutenibilidade(E) - External - Characteristic

Analísabilidade(E) - External - SubCharacteristic (1)

Tempo no entendimento das funcionalidades do siste - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas para entender as funcionalidades do sistema

Tempo na coleta de informacoes - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na obtencao de informacoes sobre o sistema (coleta de informacoes sobre o dominio da aplicacao, entrevistas com usuarios e desenvolvedores, etc.).

Tempo na analise do codigo - External - Attribute

Tempo na analise do codigo fonte de todos os modulos que compoem o sistema

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na analise do codigo fonte do sistema (soma dos tempos na analise de cada modulo do sistema)

Tempo na elaboracao do modelo de ciclo de vida - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na elaboracao do modelo de ciclo de vida do sistma

Tempo na definicao dos temas - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na definicao dos temas do sistema

Tempo na elaboracao do modelo de objetos - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na elaboracao do modelo de objetos do sistema (soma dos tempos na elaboracao do modelo de objetos para cada tema definido para o sistema)

Tempo na elaboracao do quadro de indice de procedi - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na elaboracao do quadro de indice de procedimentos do sistema

Tempo na elaboracao do quadro de chamadas de proce - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na elaboracao do quadro de chamadas de procedimentos do sistema

Tempo na elaboracao do quadro de op-procededs implem - External - Attribute

Unit: Tempo(horas) - Interval (float) - Upper Bound: 100000 - Lower Bound: 0

Counting Rule: Tempo em horas na elaboracao do quadro de operacoes-procedimentos de implementacao (interface e temas do sistema)

Projeto(E) - External - Characteristic

Tempo de projeto - External - Attribute

Unit: Tempo(meses) - Interval (float) - Upper Bound: 1000 - Lower Bound: 0

Counting Rule: Tempo requerido no projeto de engenharia reversa

Manutenibilidade(I) - Internal - Characteristic

Modularidade - Internal - SubCharacteristic (1)

Acoplamento - Internal - SubCharacteristic (2)

Fan-in - Internal - Attribute

Fan-out - Internal - Attribute

Fan-in - procedimentos - Internal - Attribute

Fan-out - procedimentos - Internal - Attribute

Simplicidade - Internal - SubCharacteristic (2)

Tamanho do sistema - loc - Internal - Attribute
Tamanho do sistema - loc executaveis - Internal - Attribute
Tamanho do módulo - loc - Internal - Attribute
Tamanho do módulo - loc executaveis - Internal - Attribute
Modulos do sistema - Internal - Attribute
Procedimentos - modulo - Internal - Attribute
Procedimentos - sistema - Internal - Attribute
Classes - modelo de objetos - Internal - Attribute
Classes - tema - Internal - Attribute
Numero de sentencas - Internal - Attribute
Numero de estados - Internal - Attribute
Numero de transicoes - Internal - Attribute
Numero de operacoes - Internal - Attribute
Procedimentos - interface - Internal - Attribute
Procedimentos - tema - Internal - Attribute
Densidade de procedimentos - interface - Internal - Attribute
Densidade de procedimentos - tema - Internal - Attribute

Analisabilidade(I) - Internal - SubCharacteristic (1)

Legibilidade do codigo - Internal - SubCharacteristic (2)

Densidade de comentarios - modulo - Internal - Attribute
Densidade de comentarios - sistema - Internal - Attribute
Linhas de comentarios - sistema - Internal - Attribute
Linhas de comentarios - modulo - Internal - Attribute

Projeto(I) - Internal - Characteristic

Pessoas alocadas ao projeto - Internal - Attribute
Nivel de conhecimento - Internal - Attribute
Data de inicio do projeto - Internal - Attribute
Data final do projeto - Internal - Attribute
Data final da fase de recuperacao de informacoes - Internal - Attribute
Data final da fase de recuperacao de visoes funcio - Internal - Attribute
Data final da fase de recuperacao de visoes estrut - Internal - Attribute
Atividade executada - Internal - Attribute

Usabilidade(I) - Internal - Characteristic

Operacionalidade - Internal - SubCharacteristic (1)

Profundidade menu - Internal - Attribute
Opcoes de menu - total - Internal - Attribute
Telas do sistema - Internal - Attribute

Portabilidade(I) - Internal - Characteristic

Adaptabilidade - Internal - SubCharacteristic (1)

Linguagem de implementacao - Internal - Attribute
Modulos dependentes de hardware - Internal - Attribute
Densidade de modulos dependentes de hardware - Internal - Attribute
Tipo de plataforma - Internal - Attribute

Apêndice C – Relatório SQUID da Integração dos Modelos de Qualidade e de Desenvolvimento

Collated Model Report

Report Type: Full structure report Report date: 10/06/01

Collated Model: MQ ER+MD ER

Collated on: 10/06/01

Review Points

Recuperacao de informacoes - NOT repeating

Final da fase de recuperacao de informacoes existentes sobre o sistema.

ASSOCIATED MEASURES:

Data final da fase de recuperacao de informacoes -

Unit: Data - Date

Counting Rule: Data final da fase de recuperacao de informacoes sobre o sistema.

Recuperacao de visoes funcionais - NOT repeating

Final da fase de recuperacao de visoes funcionais do sistema

ASSOCIATED MEASURES:

Data final da fase de recuperacao de visoes funcio -

Unit: Data - Date

Counting Rule: Data final da fase de recuperacao de visoes funcionais do sistema

Recuperacao de visoes estruturais - NOT repeating

Final da fase de recuperacao de visoes estruturais do sistema

ASSOCIATED MEASURES:

Data final da fase de recuperacao de visoes estrut -

Unit: Data - Date

Counting Rule: Data final da fase de visoes estruturais do sistema.

Gerenciamento do projeto - NOT repeating

Final da fase de gerenciamento do projeto de engenharia reversa.

ASSOCIATED MEASURES:

Data final do projeto -

Unit: Data - Date

Counting Rule: Data final do projeto de engenharia reversa.

Activities

Obter informacoes existentes sobre o sistema - NOT repeating

Obter informacoes disponiveis do sistema

ASSOCIATED MEASURES:

Atividade executada -

Unit: Executada - Nominal

Counting Rule: Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)

Elaborar modelo de ciclo de vida do sistema - NOT repeating

Elaborar o modelo de ciclo de vida do sistema a partir da interface do sistema

ASSOCIATED MEASURES:

Atividade executada -

Unit: Executada - Nominal

Counting Rule: Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)

Elaborar modelo de operacoes do sistema - NOT repeatingASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Definir temas - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Elaborar modelo de objetos do sistema - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Elaborar quadro de chamadas de procedimentos - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Elaborar quadro de indice dos procedimentos - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Elaborar quadro op-procedds de implementacao - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)**Obter informacoes sobre gerenciamento do projeto - NOT repeating**ASSOCIATED MEASURES:Atividade executada -**Unit:** Executada - Nominal**Counting Rule:** Indica se a respectiva atividade foi executada completamente (uma atividade é considerada executada quando todos os seus produtos foram gerados)

Deliverables

Informacoes sobre o sistema - executavel - NOT repeating

Informacoes obtidas sobre o sistema executavel

ASSOCIATED MEASURES:Tipo de plataforma -**Unit:** Plataforma - Nominal**Counting Rule:** Plataforma operacional na qual o sistema executa.Opcoes de menu - total -**Unit:** Numero de opcoes - Ratio (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Numero total de opcoes do menu do sistema

Telas do sistema -

Unit: Telas - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0

Counting Rule: Numero total de telas do sistema

Profundidade menn -

Unit: Numero de opcoes - Ratio (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0

Counting Rule: Nivel de profundidade do menu do sistema

Informacoes sobre o sistema - codigo fonte - NOT repeating

ASSOCIATED MEASURES:

Densidade de comentarios - sistema -

Unit: Lcom por loc(executaveis) - Ratio (float) - **Upper Bound:** 10000 - **Lower Bound:** 0

Counting Rule: Razao das linhas de comentarios com as linhas executaveis no codigo fonte do sistema

Densidade de modulos dependentes de hardware -

Unit: Modulos por modulos - Ratio (float) - **Upper Bound:** 10000 - **Lower Bound:** 0

Razao entre modulos

Counting Rule: Razao entre o numero de modulos dependentes de hardware com o numero de modulos do sistema.

Linhas de comentarios - sistema -

Unit: Lcom - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Soma das linhas de comentarios dos modulos que compoem o sistema.

Modulos do sistema -

Unit: Modulos - Interval (integer) - **Upper Bound:** 100000 - **Lower Bound:** 0

Numero de modulos

Counting Rule: Numero total de modulos do sistema.

Tamanho do sistema - loc -

Unit: Loc - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Soma das linhas de codigo dos modulos que compoem o sistema.

Tamanho do sistema - loc executaveis -

Unit: Loc(executaveis) - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Soma das linhas de codigo executaveis dos modulos do sistema.

Modulos dependentes de hardware -

Unit: Modulos - Interval (integer) - **Upper Bound:** 100000 - **Lower Bound:** 0

Numero de modulos

Counting Rule: Numero de modulos que contem implementacoes dependentes do hardware.

Informacoes sobre o sistema - modulos - repeating

Informacoes sobre os modulos que compoem o sistema

ASSOCIATED MEASURES:

Densidade de comentarios - modulo -

Unit: Lcom por loc(executaveis) - Ratio (float) - **Upper Bound:** 10000 - **Lower Bound:** 0

Counting Rule: Razao entre linhas de comentarios e linhas executaveis no codigo fonte de um modulo

Linguagem de implementacao -

Unit: Linguagem de programacao - Nominal

Counting Rule: Linguagem de programacao utilizada na implementacao de um modulo

Linhas de comentarios - modulo -

Unit: Lcom - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Linhas de comentarios de um modulo

Fan-out -

Unit: Modulos - Interval (integer) - **Upper Bound:** 100000 - **Lower Bound:** 0

Numero de modulos

Counting Rule: Numero de modulos que sao chamados pelo modulo.

Fan-in -

Unit: Modulos - Interval (integer) - **Upper Bound:** 100000 - **Lower Bound:** 0

Numero de modulos

Counting Rule: Numero de modulos chamando o modulo em observacao

Procedimentos - modulo -

Unit: Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Numero de procedimetos contidos em um modulo.

Tamanho do modulo - loc -

Unit: Loc - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0

Counting Rule: Numero de linhas codigo do modulo

Tamanho do módulo - loc executáveis -**Unit:** Loc(executaveis) - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero de linhas de código executaveis do modulo.**Modelo de ciclo de vida - NOT repeating****ASSOCIATED MEASURES:****Numero de estados -****Unit:** Estados - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Numero de estados do diagrama de estados.**Numero de sentencas -****Unit:** Sentencas - Interval (integer) - **Upper Bound:** 1000 - **Lower Bound:** 0**Counting Rule:** Numero de sentencas do modelo de ciclo de vida.**Numero de transicoes -****Unit:** Transicoes - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Numero de transicoes do diagrama de estados**Modelo de operacoes - NOT repeating****ASSOCIATED MEASURES:****Numero de operacoes -****Unit:** Operacoes - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Numero de operacoes do modelo de operacoes.**Temas do sistema - repeating****ASSOCIATED MEASURES:****Classes - tema -****Unit:** Classes - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Numero de classes de um tema do sistema**Quadro de chamadas de procedimentos - NOT repeating****ASSOCIATED MEASURES:****Fan-out - procedimentos -****Unit:** Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero de procedimentos chamados por um procedimento.**Fan-in - procedimentos -****Unit:** Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero de procedimentos chamando por um procedimento.**Quadro de indice de procedimentos - NOT repeating****ASSOCIATED MEASURES:****Procedimentos - sistema -****Unit:** Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero total de procedimentos contidos nos modulos do sistema.**Quadro op-proceeds implementacao - interface - NOT repeating****ASSOCIATED MEASURES:****Densidade de procedimentos - interface -****Unit:** Proceeds por proceeds - Ratio (float) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Razao entre procedimentos associados a interface e o numero de procedimentos do sistema.**Procedimentos - interface -****Unit:** Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero de procedimentos identificados como implementacao da interface do sistema.**Quadro op-proceeds implementacao - temas - repeating****ASSOCIATED MEASURES:****Densidade de procedimentos - tema -****Unit:** Proceeds por proceeds - Ratio (float) - **Upper Bound:** 10000 - **Lower Bound:** 0**Counting Rule:** Razao entre numero de procedimentos associados a um tema e o numero de procedimentos do sistema**Procedimentos - tema -****Unit:** Procedimentos - Interval (integer) - **Upper Bound:** 1000000 - **Lower Bound:** 0**Counting Rule:** Numero de procedimentos identificados como de implementacao de um tema.

Informacoes sobre o gerenciamento do projeto - NOT repeating

ASSOCIATED MEASURES:

Data de inicio do projeto -

Unit: Data - Date

Counting Rule: Data de inicio do projeto de engenharia reversa do sistema.

Pessoas alocadas ao projeto -

Unit: Pessoas - Interval (integer) - **Upper Bound:** 1000 - **Lower Bound:** 0

Counting Rule: Numero de pessoas responsáveis pela engenharia reversa do sistema.

Nível de conhecimento -

Unit: Nível de conhecimento - Ordinal

Counting Rule: Nível de conhecimento do pessoal responsável pela engenharia reversa em relação a(s) linguagem(ns) de implementação dos módulos do sistema.

Modelo de objetos - NOT repeating

ASSOCIATED MEASURES:

Classes - modelo de objetos -

Unit: Classes - Interval (integer) - **Upper Bound:** 10000 - **Lower Bound:** 0

Counting Rule: Soma do número de classes de cada tema definido para o sistema.

Apêndice D – Relatório SQUID das Unidades de Medidas Utilizadas

Units List Report

Report date: 10/06/01

Date units

Data

Interval (float) units

Tempo(horas)

Upper Bound: 100000 - Lower Bound: 0

Tempo(meses)

Upper Bound: 1000 - Lower Bound: 0

Interval (integer) units

Classes

Upper Bound: 10000 - Lower Bound: 0

Estados

Upper Bound: 10000 - Lower Bound: 0

Lcom

Upper Bound: 1000000 - Lower Bound: 0

Loc

Upper Bound: 1000000 - Lower Bound: 0

Loc(executaveis)

Upper Bound: 1000000 - Lower Bound: 0

Modulos

Numero de modulos

Upper Bound: 100000 - Lower Bound: 0

Operacoes

Upper Bound: 10000 - Lower Bound: 0

Pessoas

Upper Bound: 1000 - Lower Bound: 0

Procedimentos

Upper Bound: 1000000 - Lower Bound: 0

Sentencas

Upper Bound: 1000 - Lower Bound: 0

Telas

Upper Bound: 10000 - Lower Bound: 0

Transicoes

Upper Bound: 10000 - Lower Bound: 0

Ratio (float) units

Lcom por loc(executaveis)

Upper Bound: 10000 - Lower Bound: 0

Modulos por modulos

Razao entre modulos

Upper Bound: 10000 - Lower Bound: 0

Proceds por proceds

Upper Bound: 10000 - Lower Bound: 0

Nominal units

Executada

Scale Categories:

Sim

Nao

Linguagem de programacao

Scale Categories:

basic

c

clipper

cobol

fortran

pascal

outras

Plataforma

Scale Categories:

Pc/DOS

Pc/Windows 3.x

Pc/Windows 9.x-Me,NT-2000

Pc/Linux

Outras

Ordinal units

Nível de conhecimento

Scale Categories:

nenhum

pouco

medio

bom

otimo

Ratio (integer) units

Numero de opcoes

Upper Bound: 10000 - Lower Bound: 0

Apêndice E – Telas de Recursos Estatísticos Disponíveis na Ferramenta SQUID

A seguir são ilustradas algumas telas de recursos estatísticos disponíveis na ferramenta SQUID.