"Fundamentação teórica das métricas de software "

Adriano Garcia Tiago

Trabalho Final de Mestrado Profissional em Computação

Instituto de Computação
Universidade Estadual de Campinas

"Fundamentação teórica das métricas de software"

Adriano Garcia Tiago

24/02/2006

Banca Examinadora:

- **Profa. Dra. Ana Cervigni Guerra, Presidente**
  CenPRA - Centro de Pesquisa Renato Archer/ MCT
- **Profa. Dra . Silvia Maria Fonseca Silveira Massruhá**
  CNPTIA/Embrapa
- **Prof. Dr. Hans Kurt Edmund Liesenberg**
  Instituto de Computação - Unicamp

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Bibliotecária: Maria Júlia Milani Rodrigues – CRB8a / 2116

Tiago, Adriano Garcia

T43f Fundamentação teórica das métricas de software / Adriano Garcia Tiago -- Campinas, [S.P. :s.n.], 2006.

Orientadores : Ana Cervigni Guerra; Luiz Eduardo Buzato

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. Engenharia de software. 2. Software – Controle de qualidade. 3. Software - Medição. I. Guerra, Ana Cervigni. II. Buzato, Luiz Eduardo. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: Fundamentals of software metrics theory

Palavras-chave em inglês (Keywords): 1. Software engineering. 2. Software – Quality assurance. 3. Software – Measures.

Área de concentração: Engenharia de Software

Titulação: Mestre em Computação

Banca examinadora:

Profa. Dra. Silvia Maria Fonseca Silveira Massruhá (CNPTIA/Embrapa)

Prof. Dr. Hans Kurt Edmund Liesenberg (IC-UNICAMP)

Data da defesa: 24/02/2006

"Fundamentação teórica das métricas de software"

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigido e defendido por Adriano Garcia Tiago e aprovado pela Banca Examinadora.

Campinas, 24 de fevereiro de 2006.

Profa. Dra. Dra. Ana Cervigni Guerra, Presidente
(Orientador)

Prof. Dr. Luiz Eduardo Buzato
(co-orientador)

Trabalho Final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na área de Engenharia de Computação

TERMO DE APROVAÇÃO

Trabalho Final Escrito defendido e aprovado em 24 de fevereiro de 2006, pela Banca Examinadora composta pelos Professores Doutores:

Profa. Dra. Silvia Maria Fonseca Silveira Massruha
CNPTIA/ Embrapa

Prof. Dr. Hans Kurt Edmund Liesenberg
IC - UNICAMP

Profa. Dra. Ana Cervigni Guerra
CenPRA/ MCT

Dedico este trabalho exclusivamente à minha orientadora Ana Cervigni Guerra pelo seu apoio e compreensão

Agradecimentos

Agradeço a Deus pelas graças recebidas, pela saúde e proteção a mim dedicada, e pelas pessoas colocadas em minha vida.

Agradeço à minha família pela educação, pelo companheirismo, pelas lições de honestidade, humildade, solidariedade e, sobretudo, pelo grande amor recebido em todos os momentos da vida.

À minha orientadora, Ana Cervigni Guerra e à funcionária do IC, Cláudia Regina, que não me deixaram desistir.

Aos meu amigos, Leandro (Sho), Veridiana, Fabiana e Renata Mazzini pelo apoio, que foi fundamental para a conclusão deste trabalho.

Resumo

Avaliar produtos de software constitui uma atividade em que a demanda cresce significativamente, pois os usuários exigem cada vez mais por qualidade, eficiência, eficácia. Modelos e Métodos de avaliação da qualidade de processos e produtos de software têm se firmado como um valioso auxílio à obtenção de produtos de software com qualidade aprimorada e mais confiáveis. Para essa avaliação constitui fundamental ferramenta a medição dos artefatos e processos que envolvem o desenvolvimento de produtos de software. As caracteristicas do processo de desenvolvimento de software dependem de um gerenciamento efetivo, baseado em um plano de projeto definido com base em estimativas mais precisas.

Por meio do presente estudo, pesquisas sobre tamanho e estimativas de complexidade para sistemas de software são o foco na fundamentação das bases da Engenharia de Software. Descreve-se aqui, conceitos importantes para o entendimento da base fundamental para uso de métricas.

Uma Metodologia para Métricas de Qualidade de Software, baseadas no trabalho da ISO/IEC esta descrito e pretende interpretar a nova norma ISO/IEC 9126-2. Contribui para os esforços para a melhoria da qualidade de software e melhoria na execução do processo de medição, utilizando-se como primeiro passo o presente trabalho. O resultado desse trabalho é uma compilação em relação aos benefícios do uso de métricas à sua implementação no processo de desenvolvimento de software, considerando-se os aspectos tecnológicos, humanos e econômicos envolvidos em cada tópico abordado ao longo do trabalho.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 - Introdução

A Ciência da Computação, juntamente com a Engenharia de Software, tem buscado aperfeiçoar ferramentas computacionais e, a cada dia, novas abordagens do uso dos sistemas são adicionadas ao nosso cotidiano.

A dependência e demanda crescentes da sociedade em relação à Informática e, em particular, a Software, tem ressaltado uma série de problemas relacionados ao processo de desenvolvimento de software.

Estes problemas afligem a área de Software desde sua criação. Estudos identificam como uma aflição crônica, e não uma crise pontual.

Para desenvolvê-los, a comunidade de Engenharia de Software, ao longo dos últimos 30 anos, estuda e implementa práticas de desenvolvimento de software bem organizadas e documentadas. Dentre essas práticas, métricas se destaca, também pelo fato de trazer conhecimentos do processo de desenvolvimento, entre outros fatores benéficos a esse tipo de produto.

1.1 Objetivo do Trabalho

O objetivo deste trabalho consiste em descrever as bases da Engenharia de Software que sustentam a teoria das métricas, explorando o que existe na área em termos de desenvolvimento e manutenção de software. O conteúdo baseia em pesquisas das obras de autores envolvidos com a Engenharia de Software, incluindo a descrição e abstração de modelos que facilitam a melhoria da qualidade da programação. Também analisamos o que existe na norma internacional ISO/IEC 9126-2.

1.2 Limitações e Restrições

O objetivo deste trabalho é reunir a fundamentação da teoria das métricas, sem entrar no detalhe da mesma. Não faz parte do escopo implementar modelos de métricas, mas sim contribuir com um estudo e apontar as bases para trabalhos futuros sobre a evolução das métricas.

1.3 Estrutura da dissertação

Inicialmente, neste capítulo, através de uma descrição do panorama inicial, coloca-se a situação da Qualidade de Software, apresentando para compreensão, a necessidade dos esforços que vêm sendo empregados na sua melhoria.

Os Capítulos 2, fala da importância do software na nossa sociedade e como nasceu o termo Engenharia de Software. Apresentam-se ao leitor, pesquisas sobre tamanho e estimativas de complexidade para sistemas de software. No Capítulo 3, este focado na fundamentação das bases da Engenharia de Software, descreve conceitos importantes para o entendimento dos assuntos tratados ao longo do trabalho. Esta base é fundamental para entender os conceitos e a importância das métricas.

Já o Capítulo 4 detalha a teoria das métricas, incluindo alguns pontos de discussão atuais, como o papel das métricas no reuso de componentes de software. Descreve uma metodologia para métricas de Qualidade de Software, até sua implantação, análise de resultados, validação ou não segundo critérios para cada um dos casos. Neste capítulo também são apresentados alguns exemplos de métricas.

O Capítulo 5 descreve uma Metodologia para Métricas de Qualidade de Software, baseadas no trabalho da ISO/IEC detalhando a nova norma ISO/IEC 9126-2. Finalmente exemplifica um modelo de métricas.

Finalizando, apresenta-se a Conclusão e oportunidades para novos trabalhos que podem ser realizados de forma a dar continuidade

aos esforços para a melhoria da qualidade de software e melhoria na execução do processo de medição, utilizando-se como primeiro estudo o presente trabalho, seja ele genérico ou especialista.

Capítulo 2 - Software e sua importância

2.1 Introdução

Nas últimas décadas o principal desafio era desenvolver um hardware que reduzisse o custo de processamento e armazenagem de dados. Nos anos 90 com os avanços na microeletrônica resultaram em maior poder de computação a um custo cada vez mais baixo. Hoje o desafio atual é melhorar a qualidade dos produtos de software, reduzir os custos e encontrar soluções para diversos processos ainda não automatizados. O software é o mecanismo que nos possibilita aproveitar o potencial de hardware existente no mercado.

O objetivo deste capítulo é mostrar a evolução do desenvolvimento do software contextualizando a preocupação com a qualidade.

2.2 Evolução do desenvolvimento de software

O contexto em que o software foi desenvolvido está estreitamente ligado a quase cinco décadas de evolução dos sistemas computadorizados. O melhor desempenho de hardware, menor tamanho e custo mais baixo precipitaram o aparecimento de sistemas baseados em computadores mais sofisticados. Foi mudando os processadores a válvula para os dispositivos microeletrônicos que a evolução deu inicio, hoje estes dispositivos são capazes de processar 200 milhões de instruções por segundo.

O desenvolvimento de sistemas computadorizados nas últimas décadas, o hardware sofreu contínuas mudanças, enquanto o software era

visto por muitos como uma arte "secundária" para a qual haviam poucos métodos sistemáticos.

O desenvolvimento do software [Castro01] era feito virtualmente sem administração – até que os prazos começassem a se esgotar e os custos a subir abruptamente.

O hardware tinha uma posição principal, tendo o software, um projeto sob medida para cada aplicação e tinha uma distribuição relativamente limitada. Por causa desse ambiente de software personalizado, o projeto era um processo implícito realizado no cérebro de alguém, e a documentação muitas vezes não existia. Durante os primeiros anos, como mostra a Figura 2.1, aprendeu-se muito sobre a implementação de sistemas baseados em computador, mas relativamente pouco sobre engenharia de sistemas de computador ou engenharia de software.

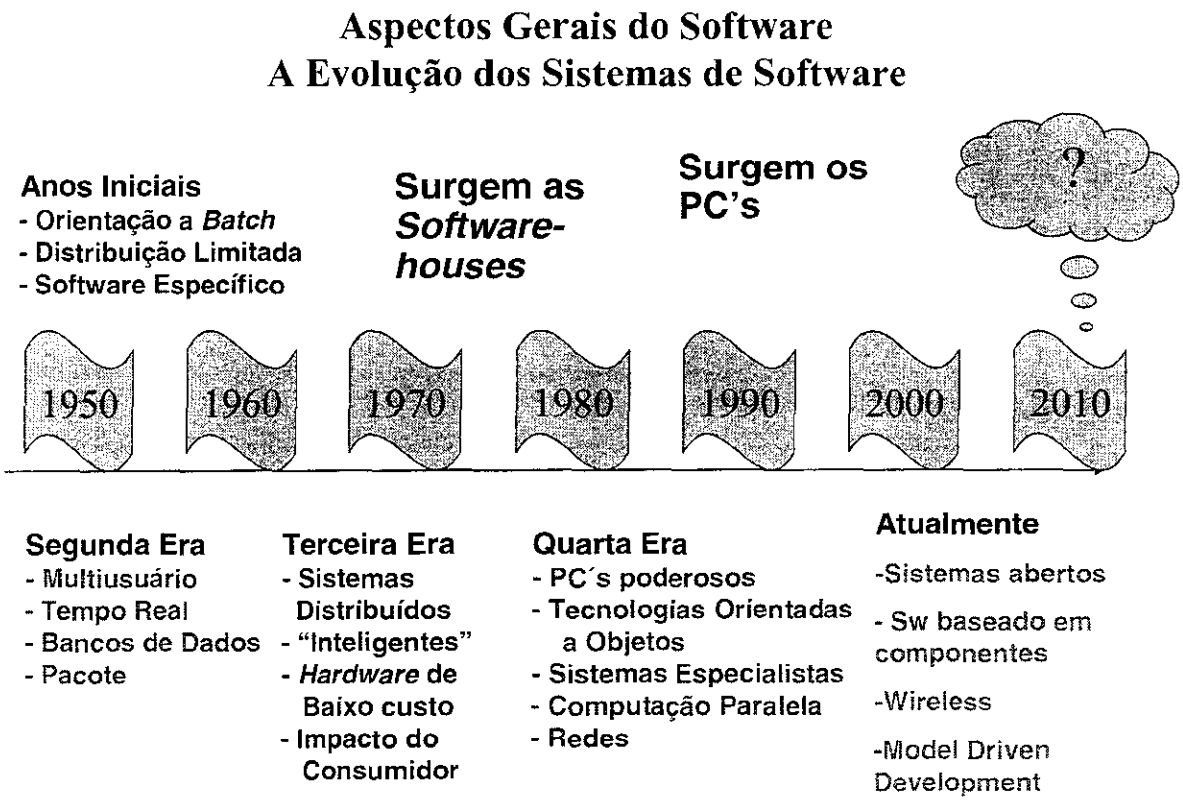

Figura 2.1- Evolução do Software

A segunda era da evolução dos sistemas computadorizados estendeu-se de meados da década de 1960 até o final da década de 1970 A multiprogramação e os sistemas multi-usuários introduziram novos conceitos de interação Homem-Máquina. As técnicas interativas abriram um novo mundo de aplicações e novos níveis de sofisticação de software e hardware. Sistemas de tempo real podiam coletar analisar e transportar dados de múltiplas fontes, daí controlando processos e produzindo saídas em milisegundos, e não em minutos. Os avanços de armazenagem *on-line* levaram a primeira geração de sistemas de gerenciamento de banco de dados. Essa evolução pode ser vista na Figura 2.1.

A segunda era também foi caracterizada pelo uso do produto de software e pelo advento das "software *houses*". O software era desenvolvido para ampla distribuição num mercado interdisciplinar. Programas para *mainframes* e microcomputadores eram distribuídos para centenas e, às vezes, milhares de usuários.

À medida que o número de sistemas baseados em computador crescia, bibliotecas de software de computador começaram a se expandir. Projetos de desenvolvimento internos nas empresas produziram dezenas de milhares de instruções de programa. Os produtos de software comprados no exterior acrescentaram centenas de novas instruções. Uma nuvem negra apareceu no horizonte. Todos esses programas – todas essas instruções – tinham de ser corrigidos quando eram detectadas falhas, alteradas conforme as exigências do usuário modificavam-se, ou eram adaptados a um novo *hardware* que fosse comprado. Essas atividades foram chamadas coletivamente de manutenção de software. O esforço despendido na manutenção de software começou a absorver recursos em índices alarmantes.

A terceira era da evolução dos sistemas computadorizados começou em meados da década de 1970. Os sistemas distribuídos – múltiplos computadores, cada um executando funções concorrentemente e comunicando-se um com o outro – aumentaram intensamente a

complexidade dos sistemas baseados em computador. As redes globais e locais, comunicações digitais de largura de bandas (*bandwidth)* elevada e a crescente demanda de acesso "instantâneo" a dados exige muito dos desenvolvedores de software.

A terceira era também foi caracterizada pelo advento e generalizado uso dos microprocessadores, computadores pessoais e poderosas estações de trabalho (*Workstations*) de mesa. O micro processador gerou um conjunto de produtos inteligentes – de automóveis e microondas, em muitos casos a tecnologia de software está sendo integrada a produtos por equipes técnicas que entendem de *hardware*, mas que freqüentemente são principiantes em desenvolvimento de software.

O computador pessoal foi o catalisador do crescimento de muitas empresas de software. O *hardware* de computador pessoal está se tornando rapidamente um produto primário, enquanto o software oferece as características capazes de se diferenciar. De fato, quando a taxa de crescimento das vendas de computadores pessoais se estabilizou em meados da década de 1980, as vendas de software continuaram a crescer. Na indústria ou em âmbito doméstico, muitas pessoas gastaram mais dinheiro em software do que aquilo que despenderam para comprar o computador no qual o software seria instalado.

A quarta era do software de computador foi caracterizada pelo surgimento e auge das tecnologias orientadas a objetos ocupando rapidamente o lugar das abordagens convencionais para o desenvolvimento de software em muitas áreas de aplicação. As técnicas de "quarta geração" para o desenvolvimento de software mudou a maneira segundo o qual alguns segmentos da comunidade de software constróem programas de computador. Os sistemas especialistas e o software de inteligência artificial finalmente saíram do laboratório para a aplicação prática em problemas de amplo espectro do mundo real.

O desenvolvimento de Software, opção de longo prazo e alto custo, ficou imprescindível. Normalmente, a metodologia de desenvolvimento adotada é a prototipação [Pressman99], privilegiando a participação do usuário (consultiva) em todo o processo. A necessidade de envolvimento do usuário aumenta conforme o tamanho e complexidade do sistema. A grande vantagem é que o programa a ser desenvolvido tem como objetiva atender ("teoricamente") a todos os requisitos (exigências) do cliente.

Com todo esse desenvolvimento e complexidade dos problemas automatizados, ninguém havia previsto a necessidade de medição apesar de [Kaplan97] ter pregado que o que não pode ser medido não pode ser gerenciado.

A estimativa de tamanho de projeto de software ainda está em evolução nas empresas brasileiras e internacionalmente existe um grupo de pesquisa, International Function Point Users Group (IFPUG) que estuda uma metodologia para estimativa de tamanho de projeto de desenvolvimento de software. Pesquisa de qualidade e produtividade no setor de software brasileiro, realizada pela Secretaria de Política de Informática do Ministério da Ciência e Tecnologia (MCT) em 2001, demonstrou que em uma amostra de 446 organizações do mercado de software brasileiro (91,5% delas são empresas privadas), apenas 30% utilizavam algum tipo de métrica para medir a produtividade e a qualidade dos processos de software. Dentre essas, 10,3% utilizam (*Line Of Codes*) LOC, 18,2% utilizavam a (Análise de Ponto de Função)APF e 6,7% utilizam outras métricas [MCT02]. Embora o índice de uso de métricas no Brasil seja baixo, Macoratti [Macoratti05] ressalta que o interesse pela APF tem crescido bastante. Sua afirmação tem como base o aumento de profissionais brasileiros certificados pelo IFPUG como Especialistas em Pontos de Função (CFPS) nos últimos três anos (dados de 2004). Enquanto, no período de 1996-2001 havia 16 CFPS, somente em 2002 e 2003 tiveram 80 certificações. De acordo com a lista de

profissionais certificados, disponibilizada na página Brasilian Function Point Users Group (BFPUG), existem até janeiro de 2004, 139 (cento e trinta e nove) especialistas certificados pelo IFPUG no Brasil.

2.3 Conclusão do Capítulo

Este capítulo apresentou como a preocupação com a qualidade foi crescendo à medida que o desenvolvimento de software foi se tornando uma prática indispensável no mundo digital.

O próximo capítulo irá detalhar o processo de desenvolvimento de software, apresentando seus elementos essenciais, dentre os quais a qualidade de software.

Capítulo 3 - Desenvolvimento de Software

3.1 Introdução

O Processo de Desenvolvimento Software é definido por Sommerville [Sommerville03] como um conjunto de atividades e resultados associados que produzem um produto de software.'

Segundo [Schach04] a Engenharia de Software é definida como uma disciplina cujo objetivo é a produção de um software sem falhas que satisfaça as necessidades do usuário, entregue no prazo e no custo acordado. Para atingir este objetivo, técnicas apropriadas devem ser usadas em todas as fases da produção do software.

Em meados dos anos 70, Schwartz [Schwartz75] já apontava como fases principais do processo de produção de um sistema de software.

- Especificação de Requisitos: tradução da necessidade ou requisito operacional para uma descrição da funcionalidade a ser executada.
- Projeto de Sistema: tradução destes requisitos em uma descrição de todos os componentes necessários para codificar o sistema.
- Programação (Codificação): produção do código que controla o sistema e realiza a computação e lógica envolvida.

- Verificação e Integração (*Checkout*): verificação da satisfação dos requisitos iniciais pelo produto produzido.

A definição moderna oferecida por Sommerville [Sommerville03] é similar; definem as atividades como Especificação, Desenvolvimento, Validação e Evolução.

Para cada fase do processo de desenvolvimento de software existe uma série de atividades que são executadas. Estas atividades constituem um conjunto mínimo para se obter um produto de software, segundo Pressman [Pressman99]:

Engenharia de Sistema: estabelecimento de uma solução geral para o problema, envolvendo questões que estão além software.

Análise de Requisitos: levantamento das necessidades do software a ser implementado. A Análise tem como objetivo produzir uma especificação de requisitos, que convencionalmente é um documento.

Especificação de Sistema: descrição funcional do sistema. Pode incluir um plano de testes para verificar adequação.

Projeto Arquitetural: onde é desenvolvido um modelo conceitual para o sistema, composto de módulos mais ou menos independentes.

Projeto de Interface: onde cada módulo tem sua interface de comunicação estudada e definida.

Projeto Detalhado: onde os módulos em si são definidos, e possivelmente traduzidos para pseudocódigo.

Segundo Pressman [Pressman99] esse conjunto mínimo de atividades tem que ser muito bem gerenciado para executar o que foi proposto com a qualidade tão desejada pelo cliente. Todas as características do software dependem de um gerenciamento do processo (contendo as atividades a cima mencionadas) de desenvolvimento de software eficiente e eficaz. Nesse gerenciamento, é essencial a adoção de normas, modelos de maturidade de processos de software, guias como NBR ISO/IEC 12.207

(ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT), 1998); ISO/IEC 9126-2; ISO/IEC TR 15.504; CMMi [1]e PMBOK[2] e de um plano de projeto efetivo, que englobe os requisitos de qualidade do produto, exigidos pelo cliente e seja baseado em estimativas precisas de tamanho, esforço, prazos e custos. O tamanho de um projeto de software é uma das primeiras estimativas a ser realizada.

3.2 O Conceito Usabilidade

Normalmente a qualidade de um software é identificada em termos de verificação, validação e aspectos de "usabilidade".

A usabilidade é reconhecida como uma questão importante de qualidade de software, da mesma forma que outros aspectos tais como funcionalidade, confiabilidade, eficiência, manutenibilidade e portabilidade. Nesse contexto, um produto de software não está limitado a possuir apenas funcionalidade, mas também a possibilitar a utilização efetiva de todos os recursos dos quais dispõe.

A "usabilidade" avalia a interação do usuário com o programa. Ela refere-se à facilidade com que o usuário utiliza o programa. Uma dificuldade na avaliação da "usabilidade" é que ela baseia-se em julgamento, opiniões e percepções dos usuários, os quais possuem experiências, conhecimentos e preferências diferenciadas, dificultando a determinação de um conjunto de critérios objetivos de avaliação [Colombo04].

A "usabilidade" como será apresentada pode ser avaliada através de entrevistas com os usuários após utilizarem o sistema, questionários com perguntas abertas ou fechadas, observação do comportamento dos usuários ao consultar o sistema e cadernos de anotações

---

[1] CMMi (Capability Maturity Model® Integration) é uma abordagem de melhoria de processo que fornecem às organizações os elementos essenciais de um processo efetivo. Este modelo foi desenvolvido pelo SEI (*Software Engineering Institute*).

[2] O Universo de Conhecimento em Gerência de Projetos (PMBOK) foi desenvolvido pelo PMI (*Project Management Institute*) e representa todo o somatório de conhecimento dentro da profissão de gerência de projetos.

ou agendas onde os usuários anotam as suas observações durante um determinado período de utilização do sistema.

3.3 Outros Conceitos

Segundo Pressman [Pressman99] a usabilidade é o conjunto de atributos que evidenciam o esforço necessário para se poder utilizar um software, bem como o julgamento individual desse uso, por um conjunto de usuários.

Usuários podem ser interpretados mais diretamente como os usuários de software interativo. Como usuários, podem ser incluídos operadores, usuário final e usuários indiretos que estão sob a influência ou dependência do uso do software.

A capacidade para uso/usabilidade deve levar em conta os vários ambientes de usuários que o software pode afetar, estes podem compreender desde a preparação para o uso até a avaliação de resultados.

Os atributos da usabilidade compreendem:

- Operacionabilidade;
- Aprendizagem;
- Compreensibilidade.

A aprendizagem evidencia o esforço do usuário para aprender a utilizar o software. A compreensibilidade evidencia o esforço do usuário para reconhecer o conceito lógico e aplicabilidade do software. Este atributo pode ser avaliado pelos seguintes critérios:

- Documentação;
- Mensagens;
- Glossário;

- *Help on-line.*

Dumas e Redish [Dumas94] acreditam que usabilidade pode apenas ser definida em termos de condições operacionais que um produto oferece. Para isto, eles consideram os seguintes aspectos:

- Foco nos usuários - deve-se conhecer, entender e trabalhar com pessoas que representam os usuários reais e potências daqueles produtos; ninguém pode substituí-los;
- Para quais tarefas o produto será usado - se as funções do sistema não se adequarem às metas dos usuários em seus ambientes de trabalho, então o produto não será usado corretamente;
- As condições dos usuários para realizar suas tarefas - as pessoas consideram um produto "fácil de usar e de aprender a usar", baseadas no tempo que elas levam para fazer o que querem, do número de passos exigidos para realização da tarefa e do sucesso que elas têm em prever a ação correta a tomar;
- Os usuários decidirão se o produto é fácil de usar - a decisão sobre a usabilidade do produto é determinada por usuários, e não por desenvolvedores.

Para avaliar a usabilidade de um sistema, é necessário questionar previamente o que está sendo usado e por quem.

A partir deste questionamento, verifica-se que a usabilidade de qualquer sistema deverá ser expressa em função de grupos específicos de usuários executando tarefas específicas.

Segundo Smith [Smith97] é conveniente observar que uma vez familiarizados com um sistema e após tê-lo considerado usável, os usuários passarão a adaptar suas tarefas cotidianas aos recursos disponibilizados pelo

sistema, adotando-o como ferramenta de auxílio à execução de suas atividades.

Os benefícios trazidos pelos testes de usabilidade podem ser divididos em duas categorias: benefícios para usuários e para companhias que desenvolvem produtos de software.

Os testes de usabilidade são realizados nas fases de pré-projeto (testes de produtos similares e de versões anteriores), desenvolvimento (testes de protótipos, de aspectos diferentes e de mudanças) e preparação (documentação, de *help* e de empacotamento).

3.4 Qualidade do Software

Apesar de não se conseguir eliminar todo o risco de se desenvolver um software com erros, o uso de técnicas que visam a garantir a qualidade do software, entre elas, o uso sistemático da atividade de teste, torna possível aumentar o nível de confiança de que se está construindo um produto conforme o desejado, evitando, assim, futuros problemas.

Segundo [Kitchenham96] os usuários tendem a avaliar a qualidade de ferramentas de software em termos de sua interação com o produto final.

O Planejamento da Qualidade é uma ação gerencial cujo objetivo é incorporar as necessidades e desejos dos clientes aos processos que fazem parte do desenvolvimento de um produto.

A qualidade percebida é agregada ao produto de software através do uso de testes realizados junto aos usuários e através da re-alimentação dos processos de Produção, Disponibilização e Evolução (PDE) de software com os resultados destes testes e do acompanhamento do usuário no seu ambiente de uso.

3.5 Engenharia, Qualidade de Processo e Produto

A engenharia pode ser vista com uma confluência de práticas artesanais, comerciais e científicas [CTI97]. Em um primeiro momento, essas vertentes visam codificar o processo de geração de um produto. Em geral, apesar da preocupação com a qualidade estar subjacente ao desenvolvimento das técnicas de produção, a necessidade da avaliação e julgamento da qualidade de produto só é explicitada em fases posteriores, quando os métodos para geração do produto já estão consolidadas.

Assim, o controle de qualidade surge como uma necessidade. No início, são feitas verificações esparsas e não sistemáticas; em seguida adotam-se técnicas e critérios bem definidos, podendo, em alguns casos, chegar à verificação de 100% dos produtos para eliminação daqueles produzidos com defeito, impedindo que eles cheguem ao usuário. A qualidade de produto, desta forma, fica com um custo elevado, seja pelo trabalho de verificação envolvido, seja pelo desperdício representado pela detecção e eliminação de partes defeituosas do software.

Como alternativa, existe a preocupação constante em melhorar o processo de produção. Adota-se o Controle Estatístico de Processo (CEP), para identificar variações no processo de forma a corrigir desvios. Para isso há necessidade de colher dados para essa finalidade.

No passo seguinte, adota-se a noção de Sistemas de Qualidade, envolvendo toda a organização no esforço pela qualidade. A seguir amplia-se o alcance do gerenciamento da qualidade, preconizando a necessidade de se fazer um projeto voltado para a qualidade, considerando a satisfação de todos os agentes envolvidos: o cliente (produto ou serviço), o acionista (resultado financeiro), os colaboradores (crescimento profissional) e comunidade. Adota-se a visão de que a qualidade se obtém principalmente durante o projeto e concepção do produto, fazendo-o mais robusto, ou seja, menos sujeito a introdução de defeitos no processo de manufatura. A Figura 2.2 mostra a evolução dos conceitos de qualidade ao longo dos anos.

Inspeção pós-produção	Avalia o produto final, depois de pronto	1900
Controle estatístico da produção	Avalia os subprodutos das etapas de produção	1940
Procedimento de produção	Avalia todo o procedimento de produção	1950
Educação das pessoas	Avalia as pessoas envolvidas no processo	1960
Otimização dos processos	Avalia e otimiza cada processo	1970
Projeto robusto	Avalia o projeto de produção	1980
Engenharia simultânea	Avalia a própria concepção do produto	1990

Figura 3.1 - Evolução dos princípios da qualidade

Como em outras disciplinas de Engenharia, a Engenharia de software tem como objetivo a melhoria da qualidade do seu produto, como propostas de modelos de desenvolvimento, métodos e técnicas para aplicação nas diversas fases do desenvolvimento do software. A avaliação da qualidade de software, nas duas visões (processo e produto), se insere nesse esforço.

Como na produção material, não cabe a dúvida quanto avaliar a se avaliar e julgar processo ou produto. As duas abordagens são necessárias e complementares. A visão de processos de software propicia uma estrutura para a harmonização das várias disciplinas da Engenharia de Software, englobando não apenas as atividades de desenvolvimento, mas todas as atividades necessárias para a sua produção.

3.6 Tecnologia da Informação

A competitividade das empresas vem apresentando profundas mudanças nas últimas décadas. Esse fato vem exigindo rápidas e contínuas adaptações na postura estratégica dessas empresas, para sobreviver e crescer nesses novos tempos de globalização da economia.

Nos países ditos do Primeiro Mundo, a tecnologia de informação tem sido considerada como um dos fatores responsáveis pelo sucesso das organizações, tanto no nível de sobrevivência, quanto no aumento da competitividade.

Assim, informações eficazes ampliam os talentos de pessoas competentes e o desenvolvimento efetivo da tecnologia, embora exigindo uma imensa habilidade, é apenas uma parte da transformação competitiva bem-sucedida. Um elemento crucial - e muito mais desafiador - está na habilidade da liderança das empresas para adaptar a organização de modo a tirar proveito da nova tecnologia.

A atual revolução tecnológica caracteriza-se não pela centralidade de conhecimentos e informação, mas pela aplicação desses conhecimentos e dessa informação para a geração de conhecimentos e de dispositivos de processamento e comunicação da informação, em um ciclo de realimentação cumulativo entre a inovação e seu uso.

Nos dois primeiros estágios, o progresso da inovação tecnológica baseou-se em aprender usando. No terceiro estágio, os usuários aprenderam a tecnologia fazendo, o que acabou resultando na reconfiguração das redes e na descoberta de novas aplicações.

O ciclo de realimentação entre a introdução de uma nova tecnologia, seus usos e seus desenvolvimentos em novos domínios torna-se muito mais rápido no novo paradigma tecnológico. Conseqüentemente, a difusão da tecnologia amplifica seu poder de forma infinita, à medida que os usuários apropriam-se dela e a redefinem.

As novas tecnologias da informação não são simplesmente ferramentas a serem aplicadas, mas processos a serem desenvolvidos. Usuários e criadores podem tornar-se a mesma coisa. Dessa forma, os usuários podem assumir o controle da tecnologia como no caso da Internet. Segue-se uma relação muito próxima entre os processos sociais de criação e manipulação de símbolos (a cultura da sociedade) e a capacidade de produzir e distribuir bens e serviços (as forças produtivas). Segundo [Castells99], pela primeira vez na história, "a mente humana é uma força direta de produção, não apenas um elemento decisivo no sistema produtivo".

Neste sentido, os contextos culturais/institucionais e a ação social intencional interagem de forma decisiva com o novo sistema

tecnológico, mas esse sistema tem sua própria lógica embutida, caracterizada pela capacidade de transformar todas as informações em um sistema comum de informação, processando-as em velocidade e capacidade cada vez maiores e com custo cada vez mais reduzido em uma rede de recuperação e distribuição potencialmente em toda parte.

Devido a sua penetrabilidade em todas as esferas da atividade humana, a revolução da tecnologia da informação terá como ponto principal, analisar a complexidade da nova economia, sociedade e cultura em formação, abordagem dos tópicos seguintes deste trabalho.

Essa opção metodológica [Castells99] não sugere que novas formas e processos sociais surgem em conseqüência de transformação tecnológica. É claro que a tecnologia não determina a sociedade. Nem a sociedade escreve o curso da transformação tecnológica, uma vez que muitos fatores, inclusive criatividade e iniciativa empreendedora, intervêm no processo de descoberta científica, inovação tecnológica e aplicações sociais, de forma que o resultado final depende de um complexo padrão interativo.

De acordo com [Castells99] "a revolução da tecnologia da informação difundiu pela cultura mais significativa de nossas sociedades o espírito libertário dos movimentos dos anos 60". No entanto, logo que se propagaram e foram apropriadas por diferentes países, várias culturas, organizações diversas e diferentes objetivos, as novas tecnologias da informação explodiram em todos os tipos de aplicações e usos que, por sua vez, produziram inovação tecnológica, acelerando a velocidade e ampliando o escopo das transformações tecnológicas, bem como diversificando suas fontes.

3.7 Engenharia da Informação

A atual revolução tecnológica caracteriza-se não pela centralidade de conhecimentos e informação, mas pela aplicação desses conhecimentos e dessa informação para a geração de conhecimentos e de

dispositivos de processamento/comunicação da informação, em um ciclo de realimentação cumulativo entre a inovação e seu uso.

Examinemos essa análise. Os usos das novas tecnologias de telecomunicações nas duas últimas décadas passaram por três estágios distintos, a automação de tarefas, as experiências de usos e a reconfiguração das aplicações [Castells99].

A Engenharia de Informação, segundo Macaulay [Macaulay96], pode ser definida como o processo sistemático de desenvolvimento de requisitos através de processos iterativos de análise do problema, de documentação das observações resultantes métricas) em uma variedade de formatos de representação e de checagem da precisão do entendimento obtido.

A revolução da tecnologia da informação [Castells99] difundiu pela cultura mais significativa de nossas sociedades o espírito libertário dos movimentos dos anos 60.

No entanto, logo que se propagaram e foram apropriadas por diferentes países, várias culturas, organizações diversas e diferentes objetivos, as novas tecnologias da informação explodiram em todos os tipos de aplicações e usos que, por sua vez, produziram inovação tecnológica, acelerando a velocidade e ampliando o escopo das transformações tecnológicas, bem como diversificando suas fontes.

O avanço da Tecnologia da Informação vem transformando as organizações. Os sistemas de informação são hoje bases para complexas estratégias de alavancagem competitiva, e não apenas ferramentas de produtividade empresarial.

A Engenharia da Informação permitiu que Sistemas passassem a ser concebidos a partir de estruturas formais e da visão do todo para o detalhe.

O desenvolvimento dos Sistemas de Informação passou a exigir a compreensão dos negócios da empresas e da estrutura organizacional [Kacuta06].

3.8 Engenharia de Software

A engenharia de software pode ser vista como um conjunto de métodos e técnicas que visam o desenvolvimento de programas de computadores com uma qualidade adequada e economicamente viáveis.

O foco principal da engenharia de software é a lógica dos processos computadorizados; ela se preocupa com as técnicas estruturadas para especificar, projetar e escrever programas [Martin91].

Em 1982 na "IEEE *COMPCON Conference*", a engenharia de software foi definida como uma combinação das seguintes atividades [Martin91]:

Metodologia de programação: programação estruturada, abordagens de projeto, análise de requisitos, técnicas de especificação.

Tecnologia de programação: Ambientes de programação, apoio para a metodologia, processamento da palavra.

Gerenciamento: Organização do projeto, planejamento do ciclo de vida.

Teoria da ciência da computação: projeto de linguagens de programação, métodos de programação, análise de algoritmos, organizações de sistemas, trocas hardware/software.

A engenharia da informação [Martin91] é um conjunto integrado de técnicas formais pelas quais modelos de empresa, modelos de dados e modelos de processos são construídos a partir de uma base de conhecimentos de grande alcance, para criar e manter sistemas de processamento de dados.

A premissa básica da engenharia da informação é que o conjunto de dados é a parte mais importante ou a parte central no processamento de dados (PD) moderno [Martin91].

A engenharia da informação é composta de:

Metodologias de dados: análise de dados, modelagem de dados e análise de entidade-relacionamento, planejamento estratégico de dados, enciclopédia de engenharia da informação.

Análise empresarial: modelagem empresarial, análise dos fatores críticos de sucesso, planejamento estratégico de sistemas de informação, projeto de recursos de informação.

Projeto de sistemas: sistemas distribuídos, bancos de dados distribuídos, redes, automação de escritórios, análise do fluxo das informações, técnicas estruturadas e sua automação, técnicas de verificação e projeto comprovadamente correto.

Construção da aplicação: Ferramentas CASE, técnicas de especificação, geradores de códigos, linguagens de quarta geração (e não procedurais), ferramentas de apoio à decisão, e manutenção.

Computação do usuário final: linguagens e ferramentas de usuário final, computadores pessoais e software, técnicas de centro de informação.

Gerenciamento de processamento de dados: administração de dados, gerenciamento da computação dos usuários, prototipagem, ciclos de vida alternativos, gerenciamento de projeto de engenharia da informação, estruturas de organização MIS (*Management Information Systems*).

Tópicos avançados: teoria das linguagens de quarta geração (e não procedurais), teoria das linguagens de especificação, sistemas baseados no conhecimento, projeto com sistemas especialistas as automação avançada em software.

A engenharia do conhecimento é o conjunto de técnicas que visam à aquisição, representação e utilização do conhecimento, bem como dos processos de raciocínio. O objeto da engenharia do conhecimento é o desenvolvimento, produção e distribuição de sistemas inteligentes feitos pelo homem.

3.9 Evolução

A revolução ocorrida na década de 70 voltou a ocorrer no final da década de 90, onde os preços dos equipamentos voltaram a cair. Um bom número de empresas, de diversos portes, está com parte de seus sistemas com um considerável nível de informatização, foi amplamente divulgado o uso da Internet com o meio de comunicação e busca maciça de informação [Oliveira97].

Surge então a necessidade de se produzir software mais atraentes dinâmicos e com alto poder de troca de informações. As técnicas oferecidas pela metodologia estruturada não eram suficientes para atender com satisfação desejada a elaboração deste tipo de aplicação. Era necessário partir para outro tipo de metodologia, que permitisse o desenvolvimento de sistemas com estas novas características. Segundo [Oliveira96] a técnica que começou a ser adotada por parte dos profissionais da área de desenvolvimento de sistemas foi a da metodologia orientada a objetos.

3.10 Tecnologia Orientada a Objetos

As técnicas orientadas a objeto (OOT) [Sanches05] provêem métodos e mecanismos para a estruturação de modelos e programação de código correspondente aos objetos (conceitos) encontrados no domínio do problema. Por exemplo, no domínio de uma aplicação de negócios, objetos podem descrever uma conta de banco ou um cliente através de atributos (número da conta, por exemplo), comportamentos (extratos de contas) e relacionamentos com outros objetos (a conta pertence a um cliente).

Os mecanismos básicos desta tecnologia são: classes, objetos, mensagens e métodos [Furlan98]:

- Um objeto é uma unidade real ou abstrata, individualizada e identificável que modela um conceito presente na realidade humana, ocupando espaço físico (mundo físico) ou lógico (na memória).

- Uma classe é a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo por compartilhar as mesmas características de atributos, operações relações e semânticas.
- Uma mensagem é uma solicitação entre objetos para invocar certa operação. Uma operação é a lógica contida uma classe que designa-lhe um comportamento).
- Um método implementa algum aspecto do comportamento do objeto. Comportamento é a forma como um objeto age e reage, em termos das suas trocas de estado e troca de mensagens.

Uma vez que objetos utilizam o princípio da abstração de dados, o encapsulamento de informação proporciona dois benefícios principais para o desenvolvimento de sistemas:

Modularidade: o código fonte para um objeto pode ser escrito e mantido independentemente do código fonte de outros objetos. Além disso, um objeto pode ser facilmente migrado para outros sistemas.

Ocultamento de informação: um objeto tem uma interface pública que os outros objetos podem utilizar para estabelecer comunicação com ele. Mas, o objeto mantém informações e métodos privados que podem ser alterados a qualquer hora sem afetar os outros objetos que dependem dele. Ou seja, não é necessário saber como o objeto é implementado para poder utilizá-lo.

Um objeto sozinho não é muito útil e geralmente ele aparece como um componente de um grande programa que contém muitos outros objetos. Através da interação destes objetos pode-se obter uma grande funcionalidade e comportamentos mais complexos.

Cada objeto criado a partir de uma classe é denominado de instância dessa classe. Uma classe provê toda a informação necessária para construir e utilizar objetos de um tipo, cada instância pertence a uma classe e uma classe pode possuir múltiplas instâncias. Devido ao fato de todas as instâncias de uma classe compartilhar as mesmas operações, qualquer diferença de respostas a mensagens aceitas por elas, é determinada pelos valores das variáveis de instância [Yourdon99].

Segundo Oliveira [Oliveira96] os métodos orientados a objetos tiveram como característica principal a uniformização dos formalismos utilizados na análise, no projeto e na implementação. Enquanto as técnicas estruturadas introduziram formalismos na fase de análise e melhoraram os da fase de projeto, a orientação a objetos uniformizou esses formalismos (funcionais e dados) na análise, no projeto e na implementação. Isso faz com que tenhamos um único modelo que é iniciado na análise, complementado no projeto e implementado na programação.

Para Yourdon [Yourdon99] a vantagens da orientação a objetos são:

- Obtenção de componentes reutilizáveis por vários sistemas de informação.
- Os objetos criados por métodos Orientação a Objetos transcendem dos programas que foram implementados pelo fato de terem valor próprio, independente de contexto. Esta abordagem traduz-se pela redução significativa de erros de projeto e programação, redução essa obtida por meio da diminuição da complexidade e atividades de programação exigidas para o desenvolvimento de um sistema.
- Em um ambiente orientado a objetos, o desenvolvedor de sistemas pode economizar tempo e esforço fazendo escolhas, a partir de uma análise de semelhanças e diferenças entre objetos, modificando/especializando um determinado objeto ou combinado objetos, para construir um novo sistema. Em outras palavras, cada desenvolvedor herda toda a especialização aplicada na construção de objetos já existentes, o que permite melhorar a qualidade de cada novo componente ou sistema a ser construído.

3.11 Conclusão do Capítulo

Este capítulo apresentou os principais elementos teóricos que constituem o processo de desenvolvimento de software. Entender tais

elementos é essencial para a contextualização das métricas e a sua importância dentro do processo de desenvolvimento.

O próximo capítulo além de explorar a teoria das métricas, irá detalhar a sua importância dentro da engenharia de software, com ênfase na qualidade do produto de software.

Capítulo 4 - Métrica de Software

4.1 Introdução

As atividades de desenvolvimento de software, quando enfocadas com objetivos profissionais, trazem à tona uma série de dificuldades que merecem particular atenção por parte do desenvolvedor. Dentre essas dificuldades estão aquelas relacionadas a estimativas de cronograma e custo de desenvolvimento. Entretanto, antes de se fazerem futuras estimativas, é preciso efetuar medições dos parâmetros obtidos no presente e verificar os parâmetros do passado.

As bases das estimativas de software são as informações obtidas através de medições dos parâmetros dos softwares produzidos anteriormente, para finalidades semelhantes. Para esta finalidade é que se desenvolveram as Métricas de Software que serão detalhadas neste capítulo.

4.2 Qualidade de produto de software

A qualidade de um produto de software [CTI97] é resultante das atividades realizadas no processo de desenvolvimento do mesmo. Avaliar a qualidade de um produto de software é verificar, através das técnicas e atividades operacionais o quanto os requisitos são atendidos. Tais requisitos, de uma maneira geral, são as expressões das necessidades, explicitados em termos quantitativos ou qualitativos, e têm por objetivo definir as características de um software, a fim de permitir o exame de sem atendimento.

Um exame sistemático exige um processo de avaliação, que seja responsável por fornecer passos a serem seguidos por quem for avaliar a qualidade do software. Para atingir este objetivo, foram criadas as normas que padronizam os requisitos da qualidade de produto de software. Neste

contexto entra a ISO (Organização Internacional de Padrões) na definição destas normas.

O órgão responsável pela normalização técnica que representa o Brasil junto à ISO é a Associação Brasileira de Normas Técnicas (ABNT), uma entidade privada, sem fins lucrativos, fundada em 1940. O ABNT/CB-21 Computadores e Processamento de Dados é o comitê técnico de normalização ao qual está vinculado o Subcomitê de Software (SC-21:10), resultante de um convênio formalizado em junho de 1992 entre a Associação Brasileira de Normas Técnicas (ABNT), a Companhia de Informática do Paraná (CELEPAR), a empresa POLO de Software S/A (Curitiba – PR) e o Instituto de Tecnologia do Paraná (TECPAR). O Subcomitê de Software (SC-21:10) responde pelas normas de engenharia, qualidade e portabilidade de software, como as de implementações padronizadas relativas a linguagens, sistemas operacionais, bancos de dados e ambientes e aplicações.

A Figura 4.1 [Queiroz00] discrimina as comissões de estudo formadas pelas duas comissões técnicas vinculadas ao *SC-21:10*.

SUBCOMITÊ DE SOFTWARE - SC-21:10			
COMISSÃO TÉCNICA		COMISSÕES DE ESTUDO	
CT-21:101	Comissão Técnica de Engenharia de Software e Portabilidade de Software	CE-21:101.01	Qualidade de Software
		CE-21:101.03	Gerência do Ciclo de Vida do Software
		CE-21:101.04	Avaliação de processos de Software
		CE-21:101.05	Ferramenta e Ambiente
		CE-21:101.06	Estimativa de Tamanho de Software (Ponto por Função)
		CE-21:101.07	Apoio aos Processos do Ciclo de Vida do Software
		CE-21:101.08	Ergonomia de Software
CT-21:102	Comissão Técnica de Linguagens, Sistemas Operacionais e Bancos de Dados	CE-21:102.01	Linguagem SQL
		CE-21:102.07	Linguagem JAVA
		CE-21:102.09	SQL Multimedia

Figura 4.1 - Comissões do SC-21:10

Através dessa comissão foi feita a tradução da norma ISO/IEC 9126 (*Information Technology - Software Quality Characteristics and Metrics*) que trata especificamente da padronização do produto de software. A versão nacional é conhecida como NBR 13596.

O modelo de qualidade ISO/IEC 9126 [ISO/IEC 9126] categoriza os atributos de qualidade de software em seis características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Por sua vez, cada uma delas é subdividida em subcaracterísticas. As características pretendem abranger todos os aspectos de qualidade de software, ou seja, deve ser possível especificar qualquer requisito de qualidade utilizando-se das seis características. A Figura 4.2 [Queiroz00] lista estas características e subcaracterísticas.

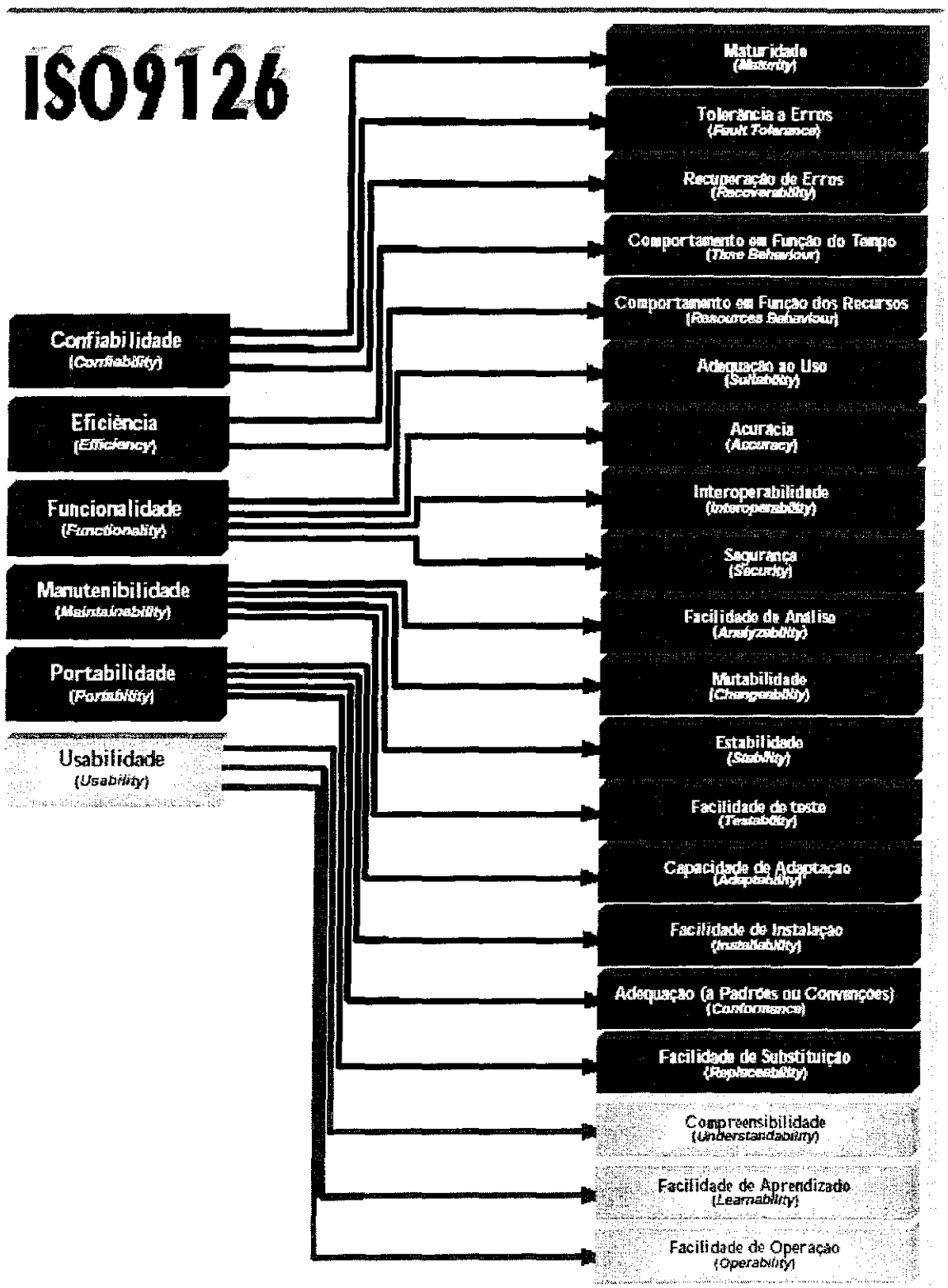

Figura 4.2 - Representação da norma ISO 9126

A Tabela 4.1 define cada característica da norma e a Tabela 4.2 fornece perguntas chave para verificar cada subcaracterística.

Característica	Definição
Confiabilidade	Conjunto de atributos relacionados com a capacidade de manutenção do grau de desempenho do produto sob condições pré-estabelecidas, por um período de tempo pré-fixado.
Eficiência	Conjunto de atributos vinculados à relação entre o desempenho do produto e a quantidade de recursos empregados, sob condições pré-estabelecidas.
Funcionalidade	Conjunto de atributos que implicam a existência de um conjunto de funções e propriedades correspondentes, cada função satisfazendo necessidades pré-estabelecidas ou subentendidas.
Manutenibilidade	Conjunto de atributos associados ao esforço necessário para a realização de alterações especificadas, incluindo desde correções, otimizações ou adaptações do produto até alterações de ambiente ou de requisitos e especificações funcionais.
Portabilidade	Conjunto de atributos associados à capacidade de migração do produto de um ambiente para outro, considerando-se os contextos organizacional e computacional (hardware e software básico)
Usabilidade	Conjunto de atributos relativos ao esforço associado ao uso do produto e à avaliação de tal uso por um universo pré estabelecido ou subentendido de usuários.

Tabela 4.1 – Descrição das características da ISO 9126

Vale a pena mencionar que o padrão ISO9126, apesar de recomendar a mensuração direta das referidas características, não sugere métricas nem indica claramente como fazê-lo. Apenas sugere que caso a característica desejada não possa ser mensurada diretamente (especialmente

durante a fase de desenvolvimento do produto), deve-se procurar medir algum outro atributo que possa auxiliar o avaliador a prognosticá-la.

Atualmente, pelo motivo acima, a norma ISO/IEC 9126 está sendo revisada. A revisão, que deverá estar pronta nos próximos anos, não deverá modificar nenhuma das características básicas da ISO/IEC 9126. A maior modificação será a inclusão de dois documentos adicionais para descrever métricas externas (relativas ao uso do produto) e métricas internas (relativas à arquitetura do produto).

Característica	**Subcaracterística**	**Pergunta chave**
Funcionalidade	Adequação	Propõe-se a fazer o que é apropriado?
	Acurácia	Faz o que foi proposto de forma correta?
	Interoperabilidade	Interage com os sistemas especificados?
	Conformidade	Está de acordo com as normas, leis, etc.?
	Segurança de Acesso	Evita acesso não autorizado aos dados?
Confiabilidade	Maturidade	Com que freqüência apresenta falha?
	Tolerância a Falhas	Ocorrendo falhas, como ele reage?
	Recuperabilidade	É capaz de recuperar dados em caso de falha?
Usabilidade	Intelegibilidade	É fácil entender o conceito e a aplicação?
	Apreensibilidade	É fácil aprender a usar? (é fácil de usar?)
	Operacionalidade	É fácil de operar e controlar?
Eficiência	Recursos	Quanto recurso usa? Durante quanto tempo?
	Tempo	Qual é o tempo de resposta, a velocidade de execução?
Manutenibilidade	Analisabilidade	É fácil de encontrar uma falha, quando ocorre?
	Modificabilidade	É fácil modificar e adaptar?
	Testabilidade	É fácil testar quando se faz alterações?
	Estabilidade	Há grande risco quando se faz alterações?

Portabilidade	Adaptabilidade	É fácil adaptar a outros ambientes?
	Capac. para ser instalado	É fácil instalar em outros ambientes?
	Conformidade	Está de acordo com padrões de portabilidade?
	Capac. Para substituir	É fácil usar para substituir outro?

Tabela 4.2 – Subcaracterísticas da ISO 9126

Um método de avaliação de produtos de software, utilizando esses conceitos e as normas aqui mencionadas, pode ser encontrado em Processo de Avaliação de Produtos de Software [GUIA05], [Colombo04], [Guerra05]. Nessas referências existe citado o Guia de Avaliação que oferece um *checklist* para uma avaliação independente de produto de software.

O objetivo dos próximos itens deste trabalho é fornecer uma base da teoria das métricas, encerrando com a descrição da norma ISO/IEC 9126-2 [ISO/IEC 9126-2], que ainda está em votação pela ABNT, que descreverá métricas externas que deixarão mais objetivas as avaliações das características de qualidade descritas na norma ISO 9126.

4.3 Introdução à métrica de software

As métricas e estimativas de software vêm se tornando um dos principais tópicos na Engenharia da Informação com a crescente exigência de seus consumidores pela qualidade, rapidez, comodidade e baixo custo de implantação e manutenção, é impossível não enxergar tais técnicas como alavanca para um produto de melhor qualidade, com custos adequados [Gomes04].

O cenário atual de alta competitividade entre as organizações faz com que a necessidade da oferta de produtos e serviços diferenciados, com qualidade e preços cada vez melhores, torne-se um fator crítico para a determinação do sucesso ou fracasso de uma organização.

Neste contexto, as empresas de desenvolvimento de software vêm se preocupando, cada vez mais, com a qualidade de seus software. Além disso, ainda que o software atinja um nível de qualidade satisfatório ao cliente, outros fatores primordiais, tais como prazos de entrega curtos e preços acessíveis, devem ser considerados.

A medição de software passa então a desempenhar um papel cada vez mais importante no entendimento e controle das práticas e produtos do desenvolvimento de software.

O conceito métrica de software refere-se à mensuração dos indicadores quantitativos do tamanho e complexidade de um sistema, sendo que estes indicadores são, por sua vez, utilizados para correlatar contra os desempenhos observados no passado a fim de derivar previsões de desempenho futuro. A métrica de software é o processo pelo qual números ou símbolos são designados a atributos de entidades do mundo real de forma a descrevê-los de acordo com regras claramente definidas.

A métrica apresenta dois indicadores importantes, a Produtividade e a Qualidade [Campos04].

De acordo com [Pressman99] o software é medido por diversas formas: indicar a qualidade do produto; avaliar a produtividade das pessoas que produzem o produto; formar uma linha básica para estimativas; ajudar a justificar os pedidos de novas ferramentas ou treinamento adicional.

Para [Gomes04] o termo métrica de software refere-se à mensuração dos indicadores quantitativos do tamanho e complexidade de um sistema. Estes indicadores são, por sua vez, utilizados para comparar com os desempenhos observados no passado a fim de derivar previsões de desempenho futuro.

A métrica de software tem como princípios especificar as funções de coleta de dados de avaliação e desempenho, atribuir essas responsabilidades a toda a equipe envolvida no projeto, reunir dados de desempenho pertencentes à complementação do software, analisar os históricos dos projetos anteriores para determinar o efeito desses fatores e

utilizar esses efeitos para pesar as previsões futuras. Estes princípios nos permitem prever o resto do processo, avaliar o progresso e reduzir a complexidade, como numa prova de *rally*, onde a cada corrida fica-se mais esclarecidos da condições e limites da equipe.

4.4 Tipos de Métricas

4.4.1 Métrica Orientada ao Tamanho

As métricas orientadas ao tamanho são medidas diretas do software e do processo por meio do qual ele é desenvolvido [Gomes04].

Segundo [Macoratti05] "as métricas de tamanho de software surgiram com o objetivo de estimar o esforço (número de pessoas-hora) e o prazo associados ao desenvolvimento de sistemas."

Na organização de software mantendo-se registros simples, uma tabela de dados orientada ao tamanho poderá ser criada. A tabela relaciona cada projeto de desenvolvimento de software que foi incluído no decorrer dos últimos anos aos correspondentes dados orientados ao tamanho deste projeto. A partir dos dados brutos contidos na tabela, um conjunto de métricas de qualidade e de produtividade orientadas ao tamanho pode ser desenvolvido para cada projeto. Médias podem ser computadas levando-se em consideração todos os projetos.

Segundo [Pressman99], as métricas orientadas ao tamanho são controversas e não são universalmente aceitas como a melhor maneira de se medir o processo de desenvolvimento de software. Grande parte dessa controvérsia diz respeito a utilização de linhas código como a chave da medida. Defensores da medição através de LOC (*Line of code*) alegam que a linha de código é o resultado de qualquer projeto de software que pode ser facilmente contado, que muitos modelos de estimativa utilizam como chave o LOC ou KLOC (*thousand lines of code*) e que existe uma vasta literatura sobre este assunto.

Por outro lado, os oponentes aclamam que as medições por linha de código são dependentes da linguagem de programação que se está

utilizando, e seu uso em estimativas requer um nível de detalhe que pode dificultar sua execução, já que o planejamento deve estimar o LOC a ser produzido antes da análise e design ter sido completadas.

4.4.2 Métricas Orientadas à Função

A métrica orientada a função [Gomes04] apresenta-se em um método para medição de software do ponto de vista do usuário, que determina de forma consistente o tamanho e complexidade de um software, sob a perspectiva do usuário.

A técnica da Análise por Pontos de Função – APF [Macoratti05] surgiu na IBM, no início da década de 70, como uma alternativa às métricas baseadas em linhas de código. Os conceitos desta técnica foram introduzidos por Allan J. Albrecth, em uma conferência do *GUIDE* – Grupo de Usuários IBM, em 1979.

> A técnica foi refinada por Albert em 1984, e, a partir desta data houve um aumento considerável na sua utilização, o que levou a necessidade de definir um padrão para aplicação da técnica. Com este objetivo foi criado em 1986 o *International Function Point Users Group* (IFPUG), já citado neste trabalho, que passou a ser responsável pela definição das regras de contagem, treinamento e certificação dos usuários da técnica. Em 1990 foi lançada a primeira versão do Manual de Práticas de Contagem ou *CPM – Counting Practices Manual* com o objetivo de padronizar a técnica.

Na a métrica orientada a função dimensiona-se um software, quantificando a funcionalidade proporcionada ao usuário a partir do seu desenho lógico. Ou seja, são medidas indiretas do software e do processo por meio do qual ele é desenvolvido. Em vez de contar linhas de código, a métrica orientada à função concentra-se na funcionalidade ou utilidade do programa. Uma abordagem foi sugerida baseada nesta proposta chamada de pontos-por-função (function point). Os pontos-por-função (FP's) são

derivados usando-se uma relação empírica baseada em medidas de informações e complexidade do software.

Segundo Simões [Simoes99] a métrica Análise de Pontos de Função é utilizada como geradora de indicadores para estimativas de prazos, gerência de recursos humanos e elaboração de planos de trabalho de projetos, assim como na avaliação e acompanhamento do progresso de projetos e análise da produtividade de equipes. A medida do tamanho de sistemas, em conjunto com tempo e custo, fornece estes indicadores, que constituem um sistema de informações gerenciais - importante ferramenta para a administração da organização.

Um outro exemplo de métrica orientada à função é o COSMIC-FPP (ISO 19761) [Cosmic06]. Esta método é uma medida de tamanho funcional que utiliza apenas o FURs (*Functional User Requirements*) do projeto de software como entrada para o processo de medição. Através dele, os requisitos funcionais podem ser decompostos em processos, e cada um desses processos funcionais é um único conjunto de subprocessos que executam movimentação e manipulação de dados. A medida do sistema é dado pela soma de todas os tipos movimentações de dados no sistema, que por convenção possuem valor 1, cada tipo.

Para Pressman [Pressman99], as métricas orientadas à função, assim como o LOC, é polêmica. Seus seguidores defendem o fato de que a análise de Pontos de Função não depende da linguagem de programação utilizada, tornando-a perfeita para aplicações que utilizam linguagens não procedurais. Alegam também que esta metodologia utiliza dados que são facilmente obtidos no início do projeto de desenvolvimento.

Já os opositores dizem que esta análise é muito subjetiva, e que não há um significado físico, sendo apenas um número. Outro ponto que levantam é que para este tipo de medição ter significância, a corporação deve estabelecer sistematicamente esse modelo de medição para que se consiga ter uma base histórica de projetos e assim uma fundamentação para as análises.

4.4.3 Métricas Orientadas a Objetos

A métrica de software também já foi estruturada para gerações passadas de tecnologia e, e hoje são usadas até para desenvolvimento Orientação a Objetos, porém não são muito coerentes, pois a diferença entre sistemas tradicionais e sistemas com orientação a objetos são muito grandes.

Existem várias propostas para métricas OO [Gomes04] que levam em consideração as características básicas e interações do sistema como: número de classes, número de cases, número de métodos, médias de métodos, médias de métodos por classe, linhas de código por método, profundidade máxima da hierarquia de classes, a relação existente entre métodos públicos e privados, entre outros.

Tais métricas baseiam-se na análise detalhada do design do sistema. Como na técnica de pontos-por-função, faz sentido adicionar um peso às métricas das classes para produzir uma medida de complexidade do sistema. A maioria das medidas examina atributos em termos dos conceitos de OO, como herança, polimorfismo e encapsulamento. Para tanto, seria necessário coletar um número significativo de contagens, ou seja, seria necessário tomar valores de vários projetos e dimensioná-los selecionando as classes, os métodos e os atributos desejáveis para medir o tamanho e a complexidade de um novo software, o que nos tomaria um longo tempo.

Segundo [Cordeiro00], assim como a análise por pontos de função, a análise orientada a objetos independe da linguagem de programação utilizada pelo sistema computacional. Por outro lado, há necessidade de uma base histórica de medições para poder se obter análises para acuradas.

4.5 Métrica e o Custo do Software

Para obter o custo de um projeto de software [Macoratti05] é preciso saber o esforço necessário para desenvolvê-lo e para determinar o esforço precisa-se saber o tamanho do projeto de software. Desta forma, determinar o tamanho de um projeto de software é uma das primeiras e

principais atividades relacionadas às estimativas a serem efetuadas durante o clico de vida do projeto.

A escolha de qual tipo de métrica utilizar vai depender muito da cultura da empresa e das características do software que será produzido. Se a empresa tem um histórico grande de projetos e medições (como por exemplo, pontos de função), é mais vantajoso utilizar a experiência passada para estimar o custo. Por outro lado, se é uma tecnologia já conhecida do mercado, pode-se utilizar a contagem por linha código, consultando *benchmarks* existentes na literatura para as estimativas.

4.6 Métricas no Gerenciamento de Projetos de Software

As métricas são ferramentas essenciais ao gerenciamento de projetos de software. A escolha das métricas está intimamente associada às estratégias e objetivos da organização, e vai depender do estágio de maturidade em que a mesma se encontra. As métricas coletadas devem prover informações que ajudem na tomada de decisões de acordo com os objetivos e estratégias da organização. Alguns desses objetivos podem ser: melhorar a qualidade do planejamento do projeto, reduzir os custos de retrabalho no processo, melhorar a qualidade do processo de desenvolvimento, melhorar a qualidade do produto resultante, reduzir os custos de falha, aumentar a produtividade do desenvolvimento, aperfeiçoar continuamente os métodos de gestão do projeto. Assim, o uso de métricas tem se tornado uma grande vantagem estratégica.

A gestão de projetos de software somente atinge determinado nível de eficácia e exatidão se houver medidas que possibilitem gerenciar através de fatos e, o que é mais importante, gerenciar os aspectos econômicos do software, que geralmente são negligenciados em organizações de desenvolvimento.

O sucesso das decisões tomadas baseadas em métricas será dependente da qualidade das métricas utilizadas. Manter a integridade dos dados e honrar a privacidade dos mesmos quando apropriada é a forma de criar um ambiente confiável, onde as pessoas irão trabalhar com maior eficiência. Nesse contexto, vale lembrar que métricas não devem ser utilizadas para medir e avaliar pessoas. Uma vez utilizadas com esse objetivo, as reportagens das métricas podem ser manipuladas e não retratar a realidade.

De acordo com o objetivo da medição, as métricas podem servir à gestão estratégica, tática e operacional de uma organização. As medições estratégicas são obtidas a partir de agregações das medições operacionais e táticas, e devem possibilitar a realização de benchmarking; melhorias contínuas no tocante à qualidade dos métodos de planejamento de projetos, gestão do processo de desenvolvimento e gestão do produto; e avaliação econômica do ativo de software. Como exemplos dessas medições podem citar: nível de satisfação do cliente, custo médio de desenvolvimento, melhoria da produtividade do desenvolvimento, benchmarking da produtividade do desenvolvimento.

As medições táticas dizem respeito à gestão do ambiente de software em termos de impacto da introdução de novas ferramentas, mudanças no processo de desenvolvimento, análise de tendências da produtividade. Alguns exemplos dessas medições são: tendência da densidade de defeitos de software, tendência da exatidão das estimativas relativas a projeto, tendência da produtividade do desenvolvimento por tipo de processo.

Já as medições operacionais, ocorrem em nível de cada projeto, visando subsidiar o planejamento de projetos, a gestão do processo de desenvolvimento, o planejamento do atendimento ao produto de software e à própria gestão do produto. Será neste tipo de medição que estaremos nos concentrando neste trabalho. Dentro das medições operacionais, ressaltamos as medições realizadas no planejamento de projetos que vão subsidiar a gestão do processo de desenvolvimento de software. Como exemplos dessas métricas podem ser citados:

- Estimativa de tamanho do software;
- Estimativa do prazo do projeto;
- Estimativa do esforço do projeto;
- Custo estimado para o projeto;
- Estimativa da distribuição de esforços por fase do projeto;
- Estimativa da distribuição de prazo por fase do projeto;
- Estimativa do número de defeitos pré-release;
- Estimativa do número de defeitos pós-release;
- Estimativa de esforço de retrabalho;
- Estimativa de custo de retrabalho;

- Estimativa do número de profissionais por fase do projeto;
- Produtividade da equipe.

A Figura 4.3 [Machado04] exemplifica várias métricas que podem ser aplicadas para se determinar a produtividade no trabalho de desenvolvimento de software. A baixa produtividade de um projeto pode afetar dois importantes itens do desenvolvimento: prazo e custo.

Nome da Métrica	Propósito da Métrica	Método de Aplicação	Medida e Fórmula	Interpretação	Tipo de Escala	Tipo de Medida	Entrada	Referência ISO 12207	Público-Alvo
Tempo da Tarefa	Quanto tempo [illegible] para completar uma tarefa?	Teste com o Usuário	X = Ta - Tb Ta = tempo acerto do usuário Tb = tempo da tarefa	X >= 0 Quanto menor, melhor	Intervalo	T = tempo	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário
Eficiência da Tarefa	Quão eficientes são os usuários?	Teste com o Usuário	X = M1 - 1 M1 = efetividade da tarefa T = tempo da tarefa	X >= 0 Quanto maior, melhor	-	T = tempo X =	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário
Custo efetivo	Qual o custo efetivo do usuário?	Teste com o Usuário	X = M1 - C M1 = efetividade da tarefa C = custo total da tarefa	X >= 0 Quanto maior, melhor	Absoluta	T = tempo X	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário
Proporção Produtiva	Que proporção do tempo o usuário está realizando ações produtivas?	Teste com o Usuário	X = Ta / Tb Ta = tempo produtivo = tempo da tarefa - tempo de ajuda - tempo perdido com erro - tempo de pesquisa Tb = tempo da tarefa	0 <= X <= 1 Quanto mais próximo de 1, melhor	Absoluta	Ta = tempo Tb = tempo X = tempo ferido	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário
Grau de Eficiência do Usuário	Quão eficiente é um usuário comparado com um especialista?	Teste com o Usuário	Grau de Eficiência do Usuário X = A / B A = eficiência de um usuário comum B = eficiência de um usuário especializado	0 <= X <= 1 Quanto mais próximo de 1, melhor	Absoluta	X = A / B	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário
Grau de Produtividade do Usuário	Quão produtivo é um usuário comparado com um especialista?	Teste com o Usuário	Grau de Produtividade do Usuário X = A / B A = produtividade de um usuário comum B = produtividade de um usuário especializado	0 <= X <= 1 Quanto mais próximo de 1 melhor	Absoluta	X = A / B	Resultado do Roteiro de Teste Monitoramento do Usuário	6.5 Validação 5.3 Teste de Qualificação 5.4 Operação	Usuários Projetistas de Interface com o Usuário

Figura 4.3 - Métricas para verificar produtividade

4.7 Métricas e o reuso de componentes

De acordo com Presman [Pressman99], componentes de software são criados através de uma série de traduções que mapeiam os requisitos do cliente com o código executável.

O reuso é uma importante característica de um componente construído com qualidade. Através dele, o componente pode ser construído e reaproveitado no desenvolvimento de outros sistemas. De acordo com Sanches [Sanches05] o reuso ganhou força na década de 70 e início dos anos 80 com o surgimento da Orientação a Objetos que trouxe o conceito de classes e objetos reusáveis. Atualmente, com o surgimento de novas tecnologias, com novas necessidades, torna-se grande a necessidade do reuso, tanto pelo custo (redução do tempo de implementação) quanto pela busca do aumento de qualidade das soluções.

Desta forma, a medição do reuso é uma forma de se provar os benefícios do mesmo. Sanches [Sanches05] alerta que o ponto de partida para qualquer programa de métricas de reuso é definir exatamente o que medir. Sem ter essa resposta plenamente consolidada o que será medido serão apenas números com pouca ou nenhuma importância.

Não existe um padrão consolidado de modelo de medição de reuso, cada organização deve adotar um modelo e usá-lo consistentemente. O autor sugere alguns pontos que devem ser considerados no momento de coleta de informações de reuso:

• "utilizado sem modificações": se o componente precisar ser modificado, então há um trabalho de desenvolvimento envolvido e isso não deve ser considerado como reuso.

• "vindo do repositório": somente componentes qualificados, documentados e desenvolvidos para reuso podem ser contados oficialmente.

• "contado apenas uma vez": assim como internamente ao projeto um componente reusado várias vezes é desenvolvido apenas uma vez, o componente vindo do repositório deve ser contado apenas uma vez.

A Figura 4.4 ilustra exemplos de métricas de reuso.

Objetivo	Métrica	
O-1	M-11	Número de componentes no repositório
	M-12	Número de acessos (sem *downloads*) ao repositório, por período
	M-13	Número de *downloads* do repositório, por período
	M-14	Número de buscas por palavras chave, sem sucesso, por período
	M-15	Número acumulado de reuso por componente
	M-16	Número acumulado de reuso para todos os componentes no repositório
O-2	M-21	Nível de reuso por projeto
	M-22	Número de acessos (sem *downloads*) ao repositório, por projeto
	M-23	Número de *downloads* do repositório, por projeto

Figura 4.4 - Métricas de Reuso de Software

As siglas "O-1" e "O-2" representam os objetivos para as métricas de reuso:

- "O-1": Caracterizar a utilização do repositório do ponto de vista do gerenciador de reuso.
- "O-2": Avaliar a utilização do reuso em um projeto do ponto de vista do gerenciador de reuso.

O conjunto de métricas ("M-11", "M12"), após coletadas e calculadas, revelam se os objetivos de reuso da organização foram cumpridos ou não.

4.8 Conclusão do Capítulo

Este capítulo apresentou os principais elementos das métricas e a sua importância dentro do contexto da qualidade de software, controle de

custo de software e do reuso. Entretanto, a métrica em si não tem sentido se não existir um processo de medição definido na corporação.

Desta forma, o próximo capítulo introduz uma metodologia para definição de um processo de medição dentro do sistema de qualidade da organização.

Capítulo 5 - Metodologia para Métricas de Qualidade de Software

5.1 Introdução

Segundo [Schulmeyer98], a definição de um processo de medição é o primeiro passo antes de se processar e analisar as métricas. Uma metodologia para métricas de qualidade de software é um abordagem sistemática para estabelecer requisitos de qualidade e de identificação, implementação, análise, e validação de métricas de qualidade para um sistema de software. Esta definição engloba cinco pontos:

- Estabelecer requisitos de qualidade de software;
- Identificar métricas de qualidade de software;
- Implementar as métricas de qualidade;
- Analisar o resultado das métricas;
- Validar as métricas de qualidade de software.

5.2 Estabelecer requisitos de qualidade de software

A metodologia inicia pela identificação dos requisitos de qualidade que podem ser aplicáveis ao sistema de software. Para definir esta lista é recomendado:

- Utilizar a experiência organizacional;
- Custos e restrições de prazo, garantias e outros pontos de interesse para a organização;
- Focar em métricas diretas ao invés de métricas de previsão. O valor de uma métrica direta é um objetivo numérico que se quer alcançar para um determinado fator ao final do desenvolvimento do produto. Já uma métrica de previsão é um objetivo numérico relacionado a um determinado fator que se quer alcançar durante o desenvolvimento do produto.

Ao terminar a lista, é necessário classificar os requisitos pela importância. A importância é dada tendo em vista as características do sistema e as opiniões das pessoas

envolvidas. Para cada requisito levantado, deve-se associar uma métrica que represente este requisito juntamente com um valor que será o objetivo a ser alcançado. Por exemplo, se a satisfação do cliente for um dos requisitos de qualidade, a métrica direta "satisfação do cliente com o produto entregue" com o valor 80% será o objetivo a ser monitorado. Esta métrica direta é usada para verificar o alcance do requisito de qualidade. Exemplos de como fazer esta medidas estão em [Colombo04].

5.3 Identificar Métricas de Qualidade de Software

Schulmayer [Schulmeyer98] recomenda o uso de um gráfico de requisitos de qualidade baseados na estrutura hierárquica mostrada na Figura 4.5. Após determinar os fatores, cada fator deve ser decomposto em subfatores até que estes estejam completos. Assim, os subfatores são decompostos em métricas usando a estrutura de métricas de qualidade.

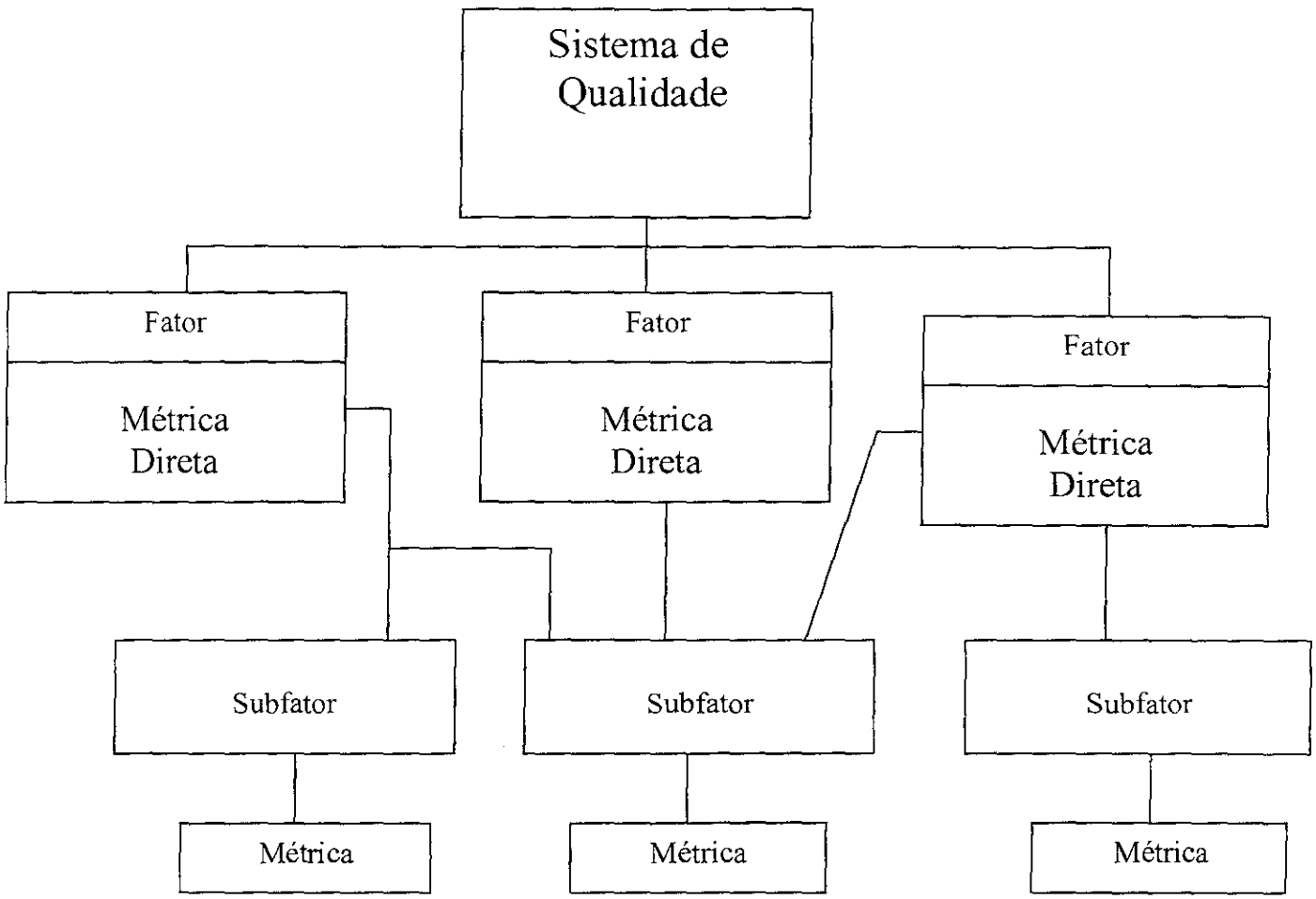

Figura 5.1 – Estrutura do sistema de métricas de qualidade

A definição da estrutura do sistema de métricas inicia com os fatores de qualidade que representam as visões macro do sistema de qualidade. Associado com cada fator está uma métrica direta, que serve como representação quantitativa do fato de qualidade. Por exemplo, uma métrica direta para o fator segurança pode ser uma falha/KLOC, que é determinado pelo gerente de projeto. Por outro lado, não há uma maneira de determinar se este fator foi alcançado.

No segundo nível da hierarquia estão os subfatores de qualidade, que representam conceitos técnicos. Eles são obtidos pela decomposição de cada fator em atributos de software mensuráveis. Os subfatores são atributos independentes de software, e podem ter correspondência com um ou mais fatores.

No terceiro nível da hierarquia os subfatores são decompostos em métricas utilizadas na medição de produtos e processos durante o ciclo de vida de desenvolvimento. Os valores de métricas diretas (fatores) são difíceis de coletar no início do desenvolvimento do produto de software, por este motivo métricas neste nível são usadas para estimar valores dos fatores.

Para cada métrica validada no terceiro nível, deve-se associar um valor que será o objetivo a ser alcançado durante o desenvolvimento. Para garantir que as métricas estejam sendo utilizadas adequadamente, apenas métricas validadas devem ser utilizadas no sistema. Métricas não validadas devem ser incluídas para futura análise, mas não devem fazer parte do sistema de requisitos.

Depois que o sistema de métricas de qualidade é aplicado, uma análise de custo-benefício deve ser feita. Para isto, é necessário identificar os custos associados à métricas do sistema. Para cada métrica, estima-se e documentam-se os seguintes impactos e custos: custos da utilização da métrica, custoso da alteração do processo de desenvolvimento de software, custos da alteração da estrutura da organização, equipamentos especiais e treinamento. Então, são inseridos para cada métrica os benefícios associados. Desta forma, forma-se o conjunto necessário para a análise custo *versus* benefícios.

5.4 Implementar as métricas de qualidade de software

Para cada métrica, determina-se se os dados que devem ser coletados e as suposições feitas sobre estes dados (por exemplo, se é uma medição objetiva ou subjetiva). Deve ser descritas também quais ferramentas deverão ser utilizadas na coleta, quais entidades da organização estarão envolvidos neste processo e quais os treinamentos necessários.

Com o processo definido, deve-se testá-lo. Validado, os dados devem ser coletados durante o ciclo de vida de desenvolvimento. O valor das métricas deve ser computado utilizando ferramentas adequadas.

5.5 Analisar o resultado das métricas

Os resultados devem ser interpretados e registrados. Componentes de software com baixo nível de qualidade são identificados. Durante o desenvolvimento as métricas validadas são usadas para fazer previsões dos valores das métricas diretas. Valores previstos das métricas diretas devem ser comparados com valores esperados para determinar se os objetivos de cada componente do software estão sendo atingidos.

Métricas diretas são utilizadas para garantir a aderência dos produtos de software com os requisitos de qualidade durante os testes de sistema e aceitação. Componentes de software e passos de processo que possuem desvios são relatados através do registro de não conformidades.

5.6 Validar as métricas de qualidade de software

Segundo Schulmeyer [Schulmeyer98] para uma métrica se tornar válida ela deve demonstrar um alto grau de associação com os fatores de qualidade que elas representam. A métrica deve ser validada seguindo certos critérios e invalidada segundo outros. Esta validação deve seguir os seguintes passos:

a. Identifique a amostra de fatores de qualidade;
b. Identifique a amostra das métricas;
c. Faça uma análise estatística;
d. Documente o resultado.

Adicionalmente, é necessário revalidar a previsão da métrica cada vez que é utilizada. A validação das métricas é um processo contínuo e o banco de métricas deve ser continuamente evoluído com novas métricas validadas. Para efeito de histórico as métricas não validadas devem continuar armazenadas, mas não devem ser mais coletadas.

5.7 Exemplo de implementação prática

Schulmeyer [Schulmeyer98] descreve um exemplo prático de implementação do sistema de qualidade pela empresa *AT&T and Bellicore*. Esta empresa implementou um conjunto de métricas que fornecem informações úteis ao projeto e à diretoria corporativa. As métricas permitem a avaliação de tendências e análise quantificativa da qualidade, iniciando com os testes de sistema. As medidas quantificam:

- O número de falhas no software, normalizado pelo tamanho do software;
- A correspondência do desenvolvimento e suporte ao cliente na resolução de problemas do cliente;
- O impacto das correções do software para o cliente.

Abaixo a descrição detalhada destas métricas:

- Densidade cumulativa de falhas internas: Falhas encontradas internamente normalizadas pelo tamanho total do sistema na fase de testes de sistema;
- Densidade cumulativa de falhas internas encontradas pelo cliente: Falhas encontradas internamente pelo cliente durante utilização do sistema normalizadas pelo tamanho do sistema entregue;
- Total de falhas críticas encontradas: Fornece o número de falhas críticas encontradas e o estado dessas falhas-abertas (não corrigidas) ou fechadas (corrigidas)-assim como o dia do registro;
- Tempo médio de fechamento das falhas críticas: Fornece a medida da correspondência do desenvolvimento e suporte ao cliente na resolução de problemas do cliente através do tempo médio que uma falha crítica permanece aberta;
- Total de correções feitas: Fornece a medida do impacto das correções feitas no sistema já entregue ao cliente.

5.8 Exemplos de métricas de qualidade

É longa a lista de características geralmente associadas à qualidade do software, como a correção, eficiência, portabilidade, confiabilidade, facilidade de utilização (usabilidade), modularidade e facilidade de manutenção. Abaixo a descrição de algumas dessas métricas.

1. Métricas de Usabilidade:

A usabilidade é definida [Abreu92] como "a capacidade segundo a qual um produto pode ser utilizado com eficiência e satisfação para atingir objetivos específicos

em ambientes específicos". Assim, ela pode ser desdobrada em cinco vertentes, cada uma com uma ponderação própria:

- Facilidade de aprendizagem - capacidade de um utilizador atingir rapidamente um grau de proficiência elevado;
- Controle - capacidade de um produto para responder de uma forma natural e consistente aos comandos e entradas de dados fornecidos;
- ·Utilidade - capacidade de resolver ou ajudar a resolver problemas para o qual o produto foi proposto;
- Eficiência - capacidade de resolver problemas de forma rápida e econômica;
- Afeto - capacidade de um produto interagir amigavelmente com os seus utilizadores.

Um dos problemas deste tipo de métricas é que as escalas não têm sentido em valor absoluto. Só podem extrair conclusões quando se comparam dois ou mais produtos. Para isso, têm de existir versões executáveis dos produtos e vários utilizadores que conheçam consistentemente as suas potencialidades.

2. Métricas de Manutenção Corretiva:

Para os utilizadores finais as falhas encontradas e a sua eficaz e eficiente eliminação são sempre o ponto mais sensível na avaliação que fazem da qualidade dos produtos que utilizam e dos processos da entidade produtora. Note-se que a noção de falha inclui também a não conformidade com a análise dos requisitos. Nesta área têm sido propostas métricas como:

- O número de falhas detectadas pela inspeção do código;
- O número de falhas detectadas por aplicação de uma bateria de testes;
- O número de falhas detectadas pelos usuários;
- O tempo médio de resposta na resolução de problemas detectados;
- O número de alterações efetuadas no código fonte.

3. Métricas de Manutenção Corretiva:

Existem várias pesquisas [Abreu92] no sentido de obter métricas de estimar o número de defeitos existentes no software e assim dimensionar o esforço de depuração (por exemplo a quantidade de testes a aplicar). Partindo das métricas de manutenção corretiva o propósito é de quantificar a confiabilidade do software. Métricas típicas deste tipo são a probabilidade de uma falha ocorrer num intervalo de tempo especificado (MTTF - "Mean Time To Fail"), ou o tempo médio entre falhas (MTBF - "Medium Time Between Failures").

Outros estudos tentam relacionar a confiabilidade com métricas de dimensão e complexidade, tais como LOC e contagem por Ponto de Função. Os trabalhos mais relevantes nesta área tentam relacionar as métricas de complexidade com a questão da manutenção. A complexidade é aí medida quer em termos internos, de cada módulo, quer em termos das inter-relações entre módulos. A correlação entre a complexidade e a manutenção pode ser assim explorada no sentido de conseguir a tão desejada redução dos custos de manutenção, através de recomendações a seguir no desenho detalhado dos sistemas.

5.9 Conclusão do capítulo

Este capítulo apresentou uma metodologia para estruturar um processo de medição e análise dentro da organização. Entretanto, se não existir um conjunto de métricas significativas que sejam capazes de mostrar a realidade do processo de desenvolvimento e de prever situações futuras, de nada adiantará um processo de medição bem definido.

O próximo capítulo descreve uma norma (ISO/IEC 9126-2) preocupada em estabelecer e normalizar um conjunto de métricas consistentes e significativas para uma parte da avaliação do produto de software.

Capítulo 6 - Métricas Externas - ISO/IEC 9126-2 - O projeto

6.1 Introdução

Embora a atual norma ISO 9126/NBR 13596 enumere as características e subcaracterísticas de um software, ela ainda não define como dar uma nota a um software em cada um destes itens. A pessoa que não está familiarizada com o processo de avaliação de software, pode ter dificuldades em tentar utilizar a norma.

Algumas características podem ser realmente medidas, como o tempo de execução de um programa, número de linhas de código, número de erros encontrados em uma sessão de teste ou o tempo médio entre falhas. Nestes casos, é possível utilizar uma técnica, uma ferramenta ou um software para realizar medições. Em outros casos, a característica é tão subjetiva que não existe nenhuma forma óbvia de medi-la.

Ficam, portanto, as questões: como dar uma nota, em valor numérico, a uma característica inteiramente subjetiva? O que representa, por exemplo, uma "nota 10" em termos de "Segurança de Acesso"? Quando se pode dizer que a "Intelegibilidade" de um software pode ser considerada "satisfatória"? Criou-se, então, uma área de estudo à parte dentro da Qualidade de Software conhecida como Métricas de Software. O que se pretende fazer é definir, de forma precisa, como medir numericamente uma determinada característica.

Para avaliar uma determinada subcaracterística subjetiva de forma simplificada, por exemplo, pode-se criar uma série de perguntas do tipo "sim ou não". Criar as perguntas de forma tal que as respostas "sim" sejam aquelas que indicam uma melhor nota para a característica. Depois de prontas as perguntas, basta avaliar o software, respondendo a cada pergunta. Se você conseguir listar 10 perguntas e o software obtiver uma resposta "sim" em 8 delas, terá obtido um valor de 80% nesta característica.

Obviamente, a técnica acima não é muito eficiente. Para melhorá-la, entretanto, pode-se garantir um número mínimo perguntas para cada característica. Além disso, algumas perguntas mais importantes podem ter pesos maiores. Estes conceitos, embora pareçam muito subjetivos, não deixam de ser uma forma eficiente de medir uma característica. Essas e outras respostas para avaliação de um produto de software estão no trabalho: Processo de Avaliação da Qualidade de Pacotes de Software, [Colombo04], [Guerra05].

Em todos os casos, um fato fica claro: nada ajuda mais a avaliar características de um software do que um avaliador experiente, que já realizou esta tarefa diversas vezes e em diversas empresas diferentes. Afinal, medir é comparar com padrões e um avaliador experiente terá maior sensibilidade do que um profissional que acaba de ler uma norma pela primeira vez. (MEDE-PRO, 1997), [GUIA05].

Atualmente, a norma ISO/IEC 9126 está sendo revisada. A revisão não modificará nenhuma das características básicas da ISO/IEC 9126. A maior modificação será a inclusão de dois documentos adicionais para descrever métricas externas (relativas ao uso do produto) e métricas internas (relativas à arquitetura do produto). A revisão, quando totalmente completada e aprovada trará:

• Algumas novas subcaracterísticas. Conformidade fará parte de todas as características. Atratividade será uma subcaracterística de usabilidade. Capacidade de coexistir será uma subcaracterística de portabilidade.
• A norma será dividida em três partes. A primeira já foi publicada (ISO/IEC 9126-1) e inclui definições e características [ISO/IEC 9126-1]. As duas seguintes descreverão métricas externas (ISO/IEC 9126-2) e internas (ISO/IEC 9126-3).
• A versão brasileira da revisão desta norma deverá ser chamada de NBR 9126-1, ISO/IEC 9126-2 e ISO/IEC 9126-3, segundo a numeração original da ISO/IEC.

Atualmente a norma ISO 9126-2 está sendo votada pela ABNT/CB-21 (Comitê Brasileiro de Computadores e Processamento de Dados) e teve origem no projeto 21:101.01-008:2003. O objetivo dos próximos itens deste trabalho é descrever esta nova norma, trazendo exemplos das métricas propostas.

6.2 A norma ISO/IEC 9126-2

Segundo ISO/IEC 9126-2 [ISO/IEC 9126-2] a avaliação do produto de software depende de técnicas de avaliação e métricas que fornecem informações sobre as características de qualidade de software. Muitas das métricas e métodos associados para a utilização dos resultados de medição podem ser usadas para avaliação de produto de

software específico. Os relatórios técnicos ISO/IEC 9126-2 e ISO/IEC 9126-3 fornecem exemplos de métricas que correspondem a uma subcaracterística.

A norma ISO/IEC 9126-2 sugere um conjunto de métricas de qualidade de software a serem utilizadas com a norma ISO/IEC 9126-1. A norma permite que o usuário modifique as métricas definidas ou utilizem novas métricas. Neste caso, é recomendado que o usuário especifique como as estas métricas se relacionam com a ISO/IEC 9126-1.

A Figura 4.6 mostra o relacionamento dos tipos de métricas definidos na norma e a sua relação com os objetivos de qualidade.

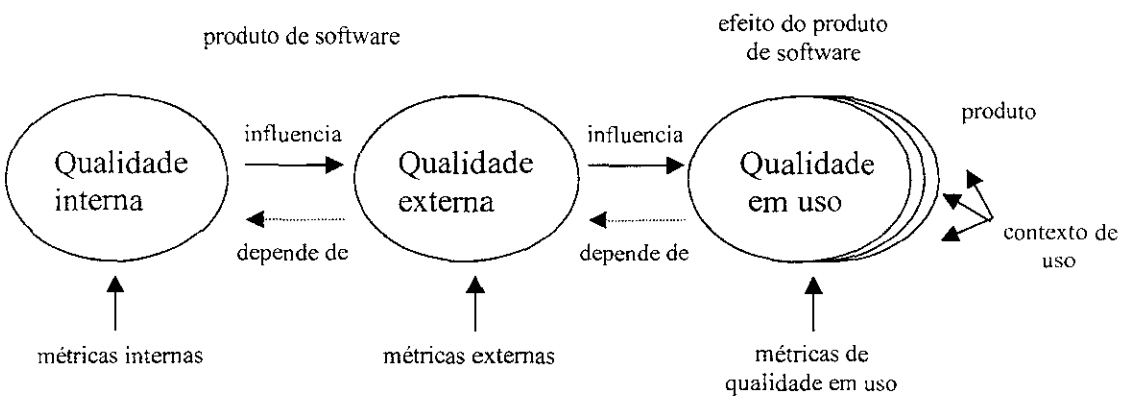

Figura 5.2 - Relacionamento entre os tipos de métricas

As métricas internas podem ser aplicadas em um produto de software não executável, durante os estágios de desenvolvimento (como pedido de proposta, definição de requisitos, especificação de projeto ou código fonte). As métricas internas permitem ao usuário medir a qualidade dos produtos intermediários e, assim, prever a qualidade do produto final. Isto possibilita que o usuário identifique problemas com a qualidade e inicie ações corretivas tão cedo quanto possível no ciclo de vida do desenvolvimento.

As métricas externas podem ser utilizadas para medir a qualidade do produto de software medindo-se o comportamento do sistema do qual o produto de software faz parte. É possível utilizar as métricas externas, no processo de ciclo de vida, durante os testes e operação. A medição é feita executando-se o produto de software no ambiente do sistema equivalente ao de operação.

As métricas de qualidade em uso permitem medir se um produto atende as necessidades de usuários especificados para que atinjam metas especificadas com eficácia,

produtividade, segurança e satisfação em contextos de uso especificados. Isto somente pode ser atingido em um ambiente de sistema equivalente ao ambiente real.

As necessidades de qualidade do usuário podem ser especificadas como requisitos de qualidade utilizando-se métricas de qualidade em uso, métricas externas e, por vezes, métricas internas. Convém que estes requisitos especificados utilizando-se métricas sejam usados como critérios de avaliação quando um produto é avaliado.

6.3 Métricas Externas Propostas

A norma lista uma série de métricas externas (vinte e sete no total) que podem ser utilizadas na avaliação do produto. Entretanto, na versão atual da norma, eles ressalvam que a lista não está finalizada podendo ainda sofrer futuras revisões.

6.3.1 Métricas de funcionalidade

Convém que uma métrica externa de funcionalidade seja capaz de medir um atributo tal como o comportamento funcional de um sistema que contenha o software. O comportamento do sistema pode ser observado a partir das seguintes perspectivas:

- Diferenças entre os resultados executados reais e a especificação de requisitos de qualidade;
- Inadequação funcional, detectada durante uma operação por usuário real, que apesar de não ser declarado, é um requisito implícito.

Observação: Quando operações ou funções implícitas são detectadas, convém que elas sejam revistas, aprovadas e estabelecidas na especificação. Convém que seja acordado o quanto elas devem ser cumpridas.

6.3.2 Métricas de adequação

Uma métrica externa de adequação deves ser capaz de medir um atributo tal como a ocorrência de uma função insatisfatória ou de uma operação insatisfatória durante o teste e durante a operação do sistema pelo usuário.

Uma função ou operação insatisfatória pode ser:

- Aquela que não executa conforme o especificado nos manuais de usuário ou na especificação de requisitos;
- Aquela que não fornece um resultado razoável e aceitável para atingir os objetivos específicos pretendidos da tarefa do usuário.

6.3.3 Métricas de acurácia

Convém que uma métrica externa de acurácia seja capaz de medir um atributo tal como a freqüência com que os usuários encontram situações de não acurácia, as quais incluem:

- Resultado incorreto ou impreciso causado por dados inadequados, por exemplo, dados com poucos dígitos significativos para um cálculo com acurácia;
- Inconsistência entre procedimentos de operação reais e os descritos no manual de operação;
- Diferenças entre os resultados reais e os esperados das tarefas executadas durante a operação.

6.3.4 Métricas de interoperabilidade

Uma métrica externa de interoperabilidade deve ser capaz de medir um atributo tal como o número de funções ou ocorrências de comunicação deficiente envolvendo dados e comandos, que deveriam ser facilmente transferidos entre o produto de software e outros sistemas, outros produtos de software ou equipamentos aos quais está conectado.

6.3.5 Métricas de segurança de acesso

Convém que uma métrica de segurança de acesso seja capaz de medir um atributo tal como o número de funções com problemas de segurança de acesso ou ocorrências de problemas de segurança de acesso, as quais são:

- Falha ao prevenir vazamento de dados ou informações seguros;
- Falha ao prevenir perda de dados importantes;
- Falha ao defender quanto ao acesso ou operação ilegal.

Observações:

- Recomenda-se que sejam executados testes de invasão para simular ataque, porque testes de ataque contra segurança de acesso normalmente não são comumente executados. Métricas reais de segurança de acesso só podem ser obtidas num "ambiente real de sistema", ou seja, "qualidade em uso";
- Requisitos de proteção referente à segurança de acesso variam muito desde um sistema isolado até um sistema conectado à Internet. A determinação da funcionalidade requerida e da garantia de sua eficácia tem sido tratada largamente em normas relacionadas ao assunto. Convém que o usuário desta norma determine funções de segurança de acesso utilizando métodos e normas apropriados, nos casos em que o impacto de qualquer dano causado seja importante ou crítico. Nos outros casos, o usuário pode limitar-se a medidas de proteção típicas de "Tecnologia da Informação (TI)" e que são geralmente aceitáveis, tais como proteção contra vírus, métodos de backup e controle de acesso.

6.3.6 Métricas de conformidade em relação à funcionalidade

Convém que uma métrica externa de conformidade em relação à funcionalidade seja capaz de medir um atributo tal como o número de funções ou ocorrências de problemas de conformidade, ou seja, produto de software não aderente a normas, convenções, contratos ou outros requisitos reguladores.

6.3.7 Métricas de confiabilidade

Uma métrica externa de confiabilidade deve ser capaz de medir atributos relacionados ao comportamento do sistema, do qual o software é uma parte, durante o teste de execução, para indicar a confiabilidade do software naquele sistema durante a operação. Na maioria dos casos, não se faz distinção entre o software e os sistemas.

6.3.8 Métricas de maturidade

Convém que uma métrica externa de maturidade seja capaz de medir atributos tais o software estar livre de falhas ocasionadas por defeitos existentes no próprio software. Aquela que não executa conforme o especificado nos manuais de usuário ou na especificação de requisitos.

6.3.9 Métricas de tolerância à falha

Convém que uma métrica externa de tolerância à falha esteja relacionada à capacidade do software de manter um nível de desempenho especificado em casos de defeitos de operação ou violação de sua interface especificada.

6.3.10 Métricas de recuperabilidade

Uma métrica externa de recuperabilidade deve ser capaz de medir atributos tais como software e sistema estarem habilitados a restabelecer um nível adequado de desempenho e recuperar os dados diretamente afetados no caso de uma falha.

6.3.11 Métricas de conformidade em relação à confiabilidade

Convém que uma métrica externa de conformidade em relação à confiabilidade seja capaz de medir atributos tais como o número de funções com problemas de conformidade ou o número de ocorrências de problemas de conformidade, quando o produto de software falha ao aderir a normas, convenções ou regulamentos relacionados à confiabilidade.

6.3.12 Métricas de Eficiência

A métrica externa de eficiência deve ser capaz de medir, durante teste e operações, atributos tais como o consumo de tempo e o comportamento de utilização de recursos do sistema computacional, incluindo software.

Recomenda-se:

- Que o tempo máximo e o tempo de distribuição sejam investigados para muitos casos de teste ou operações, porque a medida é afetada fortemente e pode variar conforme as condições de uso, tais como a carga de processamento de dados, freqüência de uso, número de locais conectados e assim por diante. Contudo, métricas de eficiência podem incluir a razão entre o valor real medido, considerando a margem de erro, e o valor projetado, considerando o limite de margem de erro permitido, conforme requerido pela especificação;
- Listar e investigar o papel desempenhado por itens tais como "UCP" e memória usada por outro software, tráfico de rede, e processos escalonados para processamento em segundo plano. Convém que limites válidos e margens de variação para valores medidos sejam estabelecidos e comparados às especificações de requisitos;
- Recomenda-se que uma tarefa seja identificada e definida como adequada para o uso do software: por exemplo, uma transação como uma tarefa para aplicação em negócios: uma comutação ou pacote de dados enviado como uma tarefa para aplicação em comunicação; um controle de evento como uma tarefa para aplicação em controle de processos; e uma saída de dados produzida por uma função a ser chamada pelo usuário para aplicação comum de usuário;

Observações:

- Tempo de resposta: tempo necessário para obter o resultado de pressionar uma chave de transmissão. Isto significa que o tempo de resposta inclui tempo de processamento e tempo de transmissão. Tempo de resposta é aplicável somente para um sistema interativo. Não existe diferença significante quando o sistema é *standalone*. Contudo, no caso de sistema de internet ou outro sistema de tempo real, algumas vezes o tempo de transmissão é muito longo;
- Tempo de processamento: o tempo decorrido em um computador entre receber uma mensagem e enviar o resultado. Algumas vezes

inclui tempo adicional de operação, outros tempos somente seria tempo usado por um programa de aplicação;

- Tempo *turn around* : tempo necessário para obter o resultado de um pedido. Em muitos casos um tempo *turn around* inclui muitas respostas. Por exemplo, no caso de um saque no caixa eletrônico de banco, o tempo *turn around* é o tempo de pressionar uma chave inicial até obter dinheiro, englobando que você precisa selecionar o tipo de transação e esperar por uma mensagem, entrar com a senha e esperar pela próxima mensagem.

6.3.13 Métricas de comportamento em relação ao tempo

Convém que uma métrica externa de comportamento em relação ao tempo seja capaz de medir tais atributos como o comportamento em relação ao tempo do sistema computacional incluindo software durante o teste ou operações.

6.3.14 Métricas de utilização de recurso

Convém que uma métrica externa de utilização de recurso seja capaz de medir tais atributos como o comportamento de utilização de recursos do sistema computacional incluindo software durante o teste ou operações.

6.3.15 Métricas de conformidade de eficiência

Uma métrica externa de conformidade de eficiência deve ser capaz de medir um atributo tal como o número de funções com, ou ocorrências de problemas de conformidade, que seja o produto de software falhando a aderência a padrões ou regulamentos relacionados a eficiência.

6.3.16 Métricas de manutenibilidade

Convém que uma métrica externa de manutenibilidade seja capaz de medir atributos, tais como, o comportamento do mantenedor, do usuário ou do sistema que contém o software, quando o software for mantido ou modificado durante os testes ou a manutenção.

6.3.17 Métricas de analisabilidade

Convém que uma métrica externa de analisabilidade seja capaz de medir atributos, tais como, o esforço do mantenedor ou do usuário ou os recursos despendidos, quando se tenta diagnosticar deficiências ou causas de falhas ou também para identificar partes a serem modificadas.

6.3.18 Métricas de modificabilidade

Uma métrica externa de modificabilidade deve ser capaz de medir atributos, tais como, o esforço do mantenedor ou do usuário medindo o comportamento do mantenedor, do usuário ou do sistema que contém o software, quando se tenta implementar uma modificação especificada.

6.3.19 Métricas de estabilidade

Convém que uma métrica externa de estabilidade seja capaz de medir atributos relacionados ao comportamento inesperado do sistema que contém o software, quando o software for testado ou operado após modificação.

6.3.20 Métricas de testabilidade

Uma métrica externa de testabilidade deve ser capaz de medir atributos, tais como, o esforço do mantenedor ou do usuário, medindo o comportamento do mantenedor, do usuário ou do sistema que contém o software, quando se tenta testar o software, modificado ou não.

6.3.21 Métricas de conformidade relacionada à manutenibilidade

Convém que uma métrica externa de conformidade relacionada à manutenibilidade seja capaz de medir um atributo, tal como, o número de funções ou de ocorrências de problemas relacionados à conformidade, em que o produto de software não adere a normas, convenções ou regulamentos requeridos, relacionados à manutenibilidade.

6.3.22 Métricas de portabilidade

Uma métrica externa de portabilidade deve ser capaz de medir atributos, tais como, o comportamento do operador ou do sistema durante a atividade de transferência.

6.3.23 Métricas de adaptabilidade

Convém que uma métrica externa de adaptabilidade seja capaz de medir atributos, tais como, o comportamento do sistema ou do usuário que está tentando adaptar o software a diferentes ambientes especificados. Quando um usuário tem que executar um procedimento de adaptação diferente do previamente fornecido pelo software, por uma necessidade de adaptação específica, convém que o esforço que é requerido ao usuário para adaptação seja medido.

6.3.24 Métricas de capacidade para ser instalado

Convém que uma métrica externa de capacidade para ser instalado seja capaz de medir atributos, tais como, o comportamento do sistema ou do usuário que está tentando instalar o software em um ambiente específico de usuário.

6.3.25 Métricas de coexistência

Uma métrica externa de coexistência deve ser capaz de medir atributos, tais como, o comportamento do sistema ou do usuário que está tentando utilizar o software com outro software independente em um ambiente comum, compartilhando recursos comuns.

6.3.26 Métricas de capacidade para substituir

Convém que uma métrica externa de capacidade para substituir seja capaz de medir atributos, tais como, o comportamento do sistema ou do usuário que está tentando utilizar o software em substituição a outro software especificado no ambiente daquele software.

6.3.27 Métricas de conformidade relacionada à portabilidade

Uma métrica externa de conformidade relacionada à portabilidade deve ser capaz de medir atributos, tais como, o número de funções ou de ocorrências de problemas

relacionados à conformidade, em que o produto de software não adere a normas, convenções ou regulamentos requeridos, relacionados à portabilidade.

6.4 Exemplo de detalhamento das métricas

A norma traz no seu texto um detalhamento de como as métricas acima podem ser utilizadas na prática. Para cada métrica geral são definidas uma ou mais métricas com uma forma de aplicação sugerida. Devido ao extenso número de tabelas com este detalhamento, será apresentada neste trabalho apenas uma, à título de exemplo.

A Tabela 5.1 mostra o detalhamento as métricas externas de analisabilidade e está estão organizada da seguinte forma:

- Nome da métrica: métricas correspondentes na tabela de métricas internas e na de métricas externas possuem nomes similares;
- Objetivo da métrica: é expresso como uma questão a ser respondida pela aplicação da métrica;
- Método de aplicação: fornece uma explicação resumida da aplicação da métrica;
- Medição, fórmula e elementos de dados para cálculos: fornece a fórmula de medição e explica o significado dos elementos de dados usados;
- Interpretação do valor medido: fornece a faixa e os valores mais adequados;
- Tipo de escala da métrica: Tipos de escalas usadas são: nominal, ordinal, intervalar, razão e absoluta;
- Tipo de medida: tipos usados são: tamanho (por exemplo, tamanho de função e tamanho de código fonte), tempo (por exemplo, tempo decorrido e tempo gasto pelo usuário), contagem (por exemplo, número de alterações e número de falhas);
- Entrada para a medição: fonte de dados utilizada na medição;
- -Referência ao PCVS da NBR ISO/IEC 12207: identifica os processos de ciclo de vida de software nos quais a métrica é aplicável;

- Público alvo: identifica os usuários dos resultados da medição.

Nome da métrica	Objetivo da métrica	Método de aplicação	Medição, fórmula e elementos de dados para cálculos	Interpretação do valor medido	Tipo de escala da métrica	Tipo de medida	Entrada para medição	Referência ao PCVS[3] da NBR ISO/IEC 12207	Público alvo
Capacidade de rastreamento das falhas	O usuário consegue identificar uma operação específica que causou a falha? O mantenedor consegue facilmente encontrar a operação específica que causou a falha?	Observar o comportamento do usuário ou do mantenedor que está tentando resolver falhas	X= A / B A= Número de dados efetivamente registrados durante a operação B= Número de dados planejados a serem registrados suficientemente para monitorar o status do software durante operação	0<=X Próximo de 1 é melhor.	Absoluta	A= Contagem B= Contagem X= Contagem/ Contagem	relatório de resolução do problema relatório de operação	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Desenvolved or Mantenedor Operador
Funções de suporte ao diagnóstico	Quão capazes são as funções de suporte ao diagnóstico ao analisar causas? O usuário consegue identificar uma operação específica que causou falha? (O usuário deve ser capaz de contornar a ocorrência da mesma falha com uma operação alternativa.) O usuário pode facilmente achar a causa da falha?	Observar o comportamento do usuário ou mantenedor que está tentando resolver falhas utilizando funções de diagnóstico.	X= A / B A= Número de falhas que o mantenedor consegue diagnosticar (utilizando a função de diagnóstico) para entender a relação causa-efeito B= Número total de falhas registradas	0<=X<= 1 Próximo de 1 é melhor.	Absoluta	A= Contagem B= Contagem X= Contagem/ Contagem	relatório de resolução do problema relatório de operação	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Desenvolved or Mantenedor Operador
Capacidade de análise das falhas	O usuário é consegue identificar a operação específica que causou a falha? O mantenedor consegue achar facilmente a causa da falha?	Observar o comportamento do usuário ou mantenedor que está tentando resolver as falhas.	X=1 - A / B A= Numero de falhas cujas causas ainda não foram encontradas B= Número total de falhas registradas	0<=X<= 1 Próximo de 1 é melhor.	Absoluta	A= Contagem B= Contagem X= Contagem/ Contagem	relatório de resolução do problema relatório de operação	5.3 Qualifica-tion testing 5.4 Operation 5.5 Mainte-nance	Desenvolved or Mantenedor Operador

[3] PCVS – Processos de Ciclo de Vida de Software
GQS – (Grupo de) Garantia da Qualidade de Software

Eficiência na análise das falhas	O usuário consegue analisar a causa de uma falha eficientemente? (O usuário algumas vezes realiza manutenção pela mudança de parâmetros.) O mantenedor consegue facilmente achar a causa de uma falha? O quão fácil é de analisar a causa de falha?	Observar o comportamento do usuário ou mantenedor que está tentando resolver as falhas.	X= Sum(T) / N T= Tout - Tin Tout = Tempo no qual a causa da falha é encontrada (ou reportada ao usuário) Tin = Tempo no qual o relatório da falha é recebido N= Número de falhas registradas	0<=X Próximo de 0 é melhor.	Razão	T= Tempo Tin,Tout= Tempo N=Contagem X= Tempo/ Contagem	relatório de resolução do problema relatório de operação	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Desenvolvedor Mantenedor Operador

NOTE: 1. É recomendado medir o tempo máximo do pior caso e a duração do tempo (bandwidth) para representar o desvio.
2. É recomendado excluir o número de falhas cujas causas ainda não foram encontradas no momento de fazer as medições. Contudo, a proporção das falhas encobertas deveria também ser medida e apresentadas conjuntamente.
3. O tempo é concernente ao ponto de vista do usuário individual, enquanto que o esforço é concernente do ponto de vista do mantenedor. Portanto, horas-homens devem ser utilizados em lugar de tempo.

Capacidade da monitoração do status	O usuário consegue identificar a operação específica que causou a falha via obtenção da informação monitorada durante a operação? O mantenedor consegue facilmente achar a causa da falha via obtenção da informação monitorada durante a operação?	Observar o comportamento do usuário ou mantenedor que está tentando obter informação monitorada de registro de status do software durante a operação.	X= 1 - A / B 7 A = Número de casos em que o mantenedor (ou usuário) não conseguiu obter informação da monitoração B = Número de casos em que o mantenedor (ou usuário) tentou obter informação da monitoração do registro de status do software durante a operação	0<=X<= 1 Melhor é próximo de 1.	Absoluta	A= Contagem B= Contagem X= Contagem/ Contagem	Relatório de resolução do problema Relatório de operação	5.3 Qualification testing 5.4 Operation 5.5 Maintenance	Usuário Desenvolvedor Mantenedor Operador

Tabela 5.1 -- Exemplo de detalhamento de métrica externa

(Fonte da Tabela: ISO/IEC 9126-2)

6.5 Considerações da norma

A norma, ainda não finalizada, traz algumas considerações interessantes para a aplicação das métricas de qualidade, como as propriedades desejáveis para um métrica, formas de utilização e um exemplo de modelo para medição da qualidade.

6.5.1 Propriedades desejáveis para métricas

Para se obter resultados válidos de uma avaliação da qualidade, convém que as métricas tenham as propriedades estabelecidas a seguir. Se uma métrica não tem estas propriedades, convém que a descrição da métrica esclareça a restrição relativa à sua validade e, se possível, como a situação pode ser contornada.

a)Confiabilidade (da métrica): A confiabilidade está associada com erro aleatório. Uma métrica é livre de erro aleatório se variações aleatórias não afetam os resultados da métrica.

b) Capacidade de Repetição (da métrica): convém que o uso repetido da métrica para o mesmo produto utilizando a mesma especificação de avaliação (incluindo o mesmo ambiente), tipo de usuários e ambiente, pelos mesmos avaliadores, produza os mesmos resultados dentro de tolerâncias apropriadas. Convém que as tolerâncias apropriadas considerem, entre outros, efeitos decorrentes de fadiga e de aprendizado.

c) Capacidade de Reprodução (da métrica): convém que o uso da métrica para o mesmo produto, utilizando a mesma especificação de avaliação (incluindo o mesmo ambiente), tipo de usuários e ambiente, por diferentes avaliadores, produza os mesmos resultados dentro de tolerâncias apropriadas.

d) Aplicabilidade (da métrica): convém que a métrica indique claramente as condições (por exemplo, a presença de atributos específicos) que restrinjam o seu uso.

e) Capacidade de Indicação (da métrica): Capacidade da métrica de indicar partes ou itens do software passíveis de melhoria, em função da comparação entre os resultados medidos e os esperados.

f) Correção (da medida): Convém que a métrica tenha as seguintes propriedades:

Objetividade (da medida): convém que os resultados da métrica e os dados de entrada sejam factuais, isto é, não influenciados por percepções ou opiniões do avaliador,

usuários de teste etc. (exceto para métricas de satisfação ou atratividade onde as percepções e opiniões de usuários são medidas).

Imparcialidade (da medida): convém que a medição não seja direcionada a nenhum resultado específico.

Precisão suficiente (da medida): a precisão é determinada pelo projeto da métrica e, particularmente, pela definição das entradas utilizadas na métrica. O usuário da métrica descreverá a precisão e sensibilidade da métrica.

g) Significância (da medida): convém que a medição produza resultados significativos sobre o comportamento ou sobre as características de qualidade do software.

Convém que a métrica também seja efetiva em relação a custo, ou seja, convém que métricas de maior custo de aplicação forneçam resultados de maior valor.

6.5.2 Demonstrando a validade de métricas

Convém que os usuários de métricas identifiquem os métodos para demonstrar a validade de métricas, como a seguir:

a)Correlação

A variação nos valores de características de qualidade (as medidas das principais métricas utilizadas), relacionada à variação dos valores obtidos pela métrica, é calculada através do quadrado do coeficiente linear.

Um avaliador pode prever características de qualidade sem medi-las diretamente utilizando métricas correlacionadas.

b)Rastreabilidade

Se uma métrica M está diretamente relacionada a um valor Q de uma característica de qualidade (as medidas das principais métricas utilizadas), para um dado produto ou processo, então uma alteração no valor Q (T1) para Q (T2), seria acompanhada por uma alteração no valor da métrica M (T1) para M (T2), no mesmo sentido (por exemplo, se Q aumenta, M aumenta).

Um avaliador pode detectar alteração dos valores das características de qualidade ao longo de um período de tempo, sem medir diretamente as características, utilizando métricas rastreáveis.

c)Consistência

Se valores de características de qualidade (as medidas das principais métricas utilizadas) Q1, Q2, ... Qn, correspondentes aos produtos ou processos 1, 2, ..., n, possuem o relacionamento Q1 > Q2 > ... > Qn, então o valores das métricas correspondentes teriam o relacionamento M1 > M2 > ... > Mn.

Um avaliador pode observar, nos componentes de software, exceções e propensão a erro utilizando métricas consistentes.

d)Previsibilidade

Se uma métrica é utilizada no tempo T1 para prever um valor Q de uma característica de qualidade (as medidas das principais métricas utilizadas) no tempo T2, então o erro de previsão é calculado por {(valor previsto Q(T2) – valor real Q(T2)) / valor real Q(T2)}. Na previsão considerar uma margem de erro dentro de limites permitidos.

Um avaliador pode prever a alteração dos valores das características de qualidade utilizando métricas que permitam previsibilidade.

e)Discriminativa

Uma métrica capaz de distinguir software de alta e baixa qualidade.

Um avaliador pode categorizar componentes de software e pontuar valores de características de qualidade utilizando métricas discriminativas.

6.5.3 Uso de métricas para estimativa (julgamento) e previsão (prognóstico)

Estimativa e previsão das características de qualidade do produto de software nos estágios iniciais são dois dos usos mais recompensadores de métricas.

1. Previsão de características de qualidade pelos dados vigentes no período da previsão

a) Previsão por análise de regressão

Ao prever o valor futuro (medida) de atributo de uma mesma característica utilizando dados vigentes deste atributo, uma análise de regressão é útil baseada em um conjunto de dados que são observados durante um período suficiente de tempo.

Por exemplo, o valor de Tempo Médio Entre Falhas (MTBF) que é obtido durante as atividades da etapa de teste pode ser utilizado para estimar o MTBF na etapa de operação.

b) Previsão por análise de correlação

Ao prever o valor futuro (medida) de atributo de uma característica utilizando dados vigentes de um atributo diferente, uma análise de correlação é útil, aplicando-se uma função de validação que demonstre a correlação.

Por exemplo, a complexidade dos módulos durante a etapa de codificação pode ser utilizada para prever o tempo ou esforço requerido para a modificação e teste do programa durante o processo de manutenção.

2. Estimativa de características de qualidade vigentes baseada em fatos vigentes

a) Estimativa por análise de correlação

Ao estimar os valores vigentes de um atributo que não seja mensurável diretamente, ou se há qualquer outra medida que possua forte correlação com a medida-alvo, uma análise de correlação é útil.

Por exemplo, em virtude de que a quantidade de defeitos residuais em um produto de software não é mensurável, ela pode ser estimada utilizando-se a quantidade e a tendência de defeitos detectados.

Convém que aquelas métricas que são utilizadas para prever os atributos que não são diretamente mensuráveis, sejam estimadas conforme explicação a seguir:

- Utilizando modelos para prever o atributo;
- Utilizando fórmula para prever o atributo;
- Utilizando a experiência para prever o atributo;
- Justificando a previsão do atributo.

Aquelas métricas que são utilizadas para prever os atributos que não são diretamente mensuráveis podem ser validadas conforme explicação a seguir:

- Identificar medidas dos atributos a serem previstos;
- Identificar as métricas a serem utilizadas para a previsão;

- Executar uma validação baseada em análise estatística;
- Documentar os resultados;
- Repetir periodicamente os itens acima.

6.5.4 Detectando desvios e anomalias em componentes propensos a apresentar problema de qualidade

As seguintes ferramentas de controle de qualidade podem ser utilizadas para analisar desvios e anomalias em componentes de produto de software:

- gráficos de processo (módulos funcionais de software)
- análise e diagramas de Pareto
- histogramas e diagramas de dispersão
- diagramas de execução, diagramas de correlação e estratificação
- diagramas de Ishikawa (espinha de peixe)
- controle estatístico de processo (módulos funcionais de software)
- folha de apontamentos

As ferramentas acima podem ser usadas para identificar questões de qualidade a partir dos dados obtidos pela aplicação de métricas.

6.5.5 Apresentando resultados da medição

a) Apresentando resultados de avaliação das características de qualidade

As seguintes representações gráficas são úteis para apresentar os resultados da avaliação da qualidade para cada característica e subcaracterística de qualidade: Gráfico de radar, histograma, gráfico de multi-variável, matriz de desempenho e importância.

b) Apresentando medidas

Existem algumas apresentações gráficas úteis tais como gráfico de Pareto, gráfico de tendência, histogramas, e gráfico de correlação.

6.6 Exemplo de um modelo de métricas

Este exemplo de estrutura é uma descrição em alto nível de como o modelo de qualidade e métricas relacionadas da série ISO/IEC 9126 podem ser utilizados durante o desenvolvimento e implementação de software para atingir uma qualidade de produto que satisfaça os requisitos especificados pelo usuário. Os conceitos mostrados neste exemplo podem ser implementados em várias formas de adaptação para atender o indivíduo, organização ou projeto. O exemplo utiliza os processos chave de ciclo de vida estabelecidos pela NBR ISO/IEC 12207 como referência para o ciclo de vida de desenvolvimento de software e os passos do processo de avaliação da qualidade da NBR ISO/IEC 14598-3 como referência ao processo de avaliação da qualidade de produto de software. Os conceitos podem estar mapeados em outros modelos de ciclo de vida de software caso o usuário deseje assim, desde que estes conceitos sejam entendidos.

6.6.1 Visão geral do processo de desenvolvimento e qualidade

A Tabela 5.2 mostra um exemplo de modelo que relaciona as atividades do processo de ciclo de vida de desenvolvimento de software (atividades de 1 a 8) a seus resultados chave e também aos modelos de referência relevantes para medição da qualidade dos resultados (isto é, qualidade interna, qualidade externa e qualidade em uso).

A linha 1 descreve as atividades do processo de ciclo de vida de desenvolvimento de software (isto pode ser adaptado para atender necessidades individuais). A linha 2 descreve se é possível obter uma medida real ou uma previsão para a categoria da medida (isto é, qualidade interna, qualidade externa e qualidade em uso). A linha 3 descreve os resultados chave que podem ser medidos para qualidade e a linha 4 descreve as métricas que podem ser aplicadas sobre cada resultado de cada atividade de processo.

	Atividade 1	**Atividade 2**	**Atividade 3**	**Atividade 4**	**Atividade 5**	**Atividade 6**	**Atividade 7**	**Atividade 8**
Fase	Análise de	Projeto da	Projeto	Codificaçã	Integração	Integração	Instalação	Apoio à

	requisitos (software e sistema)	arquitetura (software e sistema)	detalhado do software	o e teste do software	do software e teste de qualificaçã o do software	do sistema e teste de qualificaçã o do sistema	do software	aceitação do software
Modelo de referência da série NBR ISO/IEC 9126	Qualidade de usuário requerida. Qualidade interna requerida. Qualidade externa requerida.	Qualidade em uso prevista. Qualidade externa prevista. Qualidade interna medida.	Qualidade em uso prevista. Qualidade externa prevista. Qualidade interna medida.	Qualidade em uso prevista. Qualidade externa medida. Qualidade externa prevista. Qualidade interna medida.	Qualidade em uso prevista. Qualidade externa medida. Qualidade externa prevista. Qualidade interna medida.	Qualidade em uso prevista. Qualidade externa medida. Qualidade interna medida.	Qualidade em uso prevista. Qualidade externa medida. Qualidade interna medida.	Qualidade em uso medida. Qualidade externa medida. Qualidade interna medida.
Resultados chave da atividade	Requisitos de qualidade de usuário (especifica dos). Requisitos de qualidade externa (especifica dos). Requisitos de qualidade interna (especifica dos).	Projeto de arquitetura de software e de sistema.	Projeto detalhado de software.	Resultados do teste e codificação do software.	Produto de software e resultados do teste.	Sistema integrado e resultados do teste.	Sistema instalado.	Produto de software entregue.
Métrica utilizada para medir	Métricas internas (métricas	Métricas internas.	Métricas internas.	Métricas internas e externas.	Métricas internas e externas.	Métricas internas e externas.	Métricas internas e externas.	Métricas internas, externas e

	externas podem ser aplicadas para validar a especificação).							de qualidade em uso.

Tabela 5.2 – Modelo de medição da qualidade

6.7 Conclusão do Capítulo

Este capítulo apresentou detalhes da norma que pretende dar uma diretiva mais concreta da utilização das métricas na avaliação de produtos de software. Não foi escopo deste trabalho fazer a experimentação prática destas métricas, mas conforme pode ser visto na conclusão do trabalho é uma possibilidade de trabalho futuro que traria ganhos para a popularização desta norma.

Conclusão

A evolução dos sistemas de informação, juntamente com a estrutura de programação de software, fez com que se repensasse os aspectos que facilitam o processo de criação e modificação dos sistemas.

Atualmente busca-se a qualidade e a perfeita utilização do software buscando a satisfação dos usuários.

Baseado nos conceitos de objetos e classes, a programação orientada a objeto representa uma evolução no desenvolvimento software, trazendo inúmeros benefícios à criação de programas, dentre os quais o principal é a reutilização de código, que reduz drasticamente os tempos de desenvolvimento e manutenção de programas. Vale ressaltar, que as técnicas orientadas a objetos não garantem o sucesso do reuso de componentes. Segundo [Sanches05], a orientação a objeto é uma técnica que influencia positivamente o reuso, mas existem fatores (organizacionais, culturais e políticas) que determinam porque um componente de software pode nunca ser reusado. O uso de métricas vai apontar esse tipo de ocorrência para melhorar os processos de modo global, mostrando e dando evidências objetivas a respeito do modo de desenvolvimento de software e propiciando melhoria na qualidade dos produtos.

Assim a utilização da programação orientada a objetos, tende aumentar a qualidade dos produtos de software e principalmente a redução de custos em sua produção, o que certamente ajudará na popularização da informática.

A programação Orientada a Objeto não é definitiva e completa, mas é uma ferramenta que deve ser utilizada nos processos de criação de software, até que a evolução da programação traga novos elementos facilitadores para a melhoria dos processos.

Neste contexto de busca contínua de qualidade e previsibilidade, entram as métricas. O gerenciamento do projeto de software representa a primeira camada do processo de Engenharia de Software. A gerência de projetos é composta de atividades que incluem medição, análise de riscos, cronograma e controle. Assim, as métricas permitem que os gerentes entendam melhor o processo de desenvolvimento e o produto que se está produzindo.

Já as métricas de qualidade, tais como produtividade, foca tanto em processo quanto em produto. Através da análise das métricas, uma organização pode agir corretamente nas áreas do processo de Engenharia de Software que estão causando defeitos.

Entretanto, não existem hoje modelos de métricas bem definidos, o que dificulta a sua implantação efetiva em uma organização. Neste contexto, entra a norma ISO/IEC 9126-2 que tem como objetivo normalizar um conjunto de métricas único que podem ser aplicados por qualquer empresa.

Este trabalho sugere também um modelo de processo de medição e análise que pode ser implementada por uma organização. A experiência prática mostra que a utilização de uma metodologia bem definida é extremamente essencial para se alcançar os objetivos de benefícios das métricas.

Deve-se em um primeiro momento selecionar um conjunto de métricas que sejam representativas para os processos que se deseja avaliar. Após sucessivas medições e análises deve-se verificar se realmente as métricas escolhidas representam a "fotografia" do processo que se está querendo avaliar. Dessa forma, retiram-se ou acrescentam-se novas medidas e métricas de forma controlada no modelo de qualidade da organização.

O ponto mais importante é montar uma base histórica de métricas dos projetos, para se passar para o último nível de melhoria, que é análise estatística das métricas armazenadas. Através desta

análise, consegue-se construir modelos estatísticos capazes de mostrar a capacidade dos processos da organização, prever comportamentos futuros e mostrar pontos de melhoria na produção do produto.

Considera-se, assim ter atingindo o objetivo de fundamentação teórica das métricas, além de citar os pontos mais importantes dessa teoria, dentro do contexto da Engenharia de Software.

Vale lembrar, finalmente, que métricas não devem ser utilizadas para medir e avaliar pessoas. Existem outras maneiras para esse propósito. Uma vez utilizadas com esse objetivo, podem ser manipuladas e não retratar a realidade.

Proposta de trabalho futuro

Pelo pós e contras na utilização dos tipos de métricas, é interessante que se desenvolvam modelos de sistemas de métricas para determinados objetivos. Alguns exemplos:

- Sistemas de métricas para projetos CMMi;
- Sistemas de métricas para projetos que utilizam o PMI;
- Sistemas de métricas para projetos com desenvolvimento Java;

Esta classificação auxiliaria as instituições na escolha de qual modelo seguir, ajudaria no desenvolvimento contínuo destes modelos.

Um outro trabalho interessante seria fazer uma experimentação prática das métricas externas propostas pela norma ISO/IEC 9126-2. Um estudo da factíbilidade das métricas para qualquer corporação adotando as métricas propostas, analisando em termos de custos e benefícios da extração e análise destas métricas.

Como pode ser visualizado este assunto é interessante e inesgotável.

Referências Bibliográficas

[Abreu92] Abreu, Fernando Brito. **As Métricas na Gestão de Projetos de Desenvolvimento de Sistemas de Informação.** Artigo publicado na 6ª jornada para a Qualidade de Software, Lisboa 1992.

[Campos04] Campos, Fábio Martinho. **Métricas de software como ferramenta de apoio ao gerenciamento de projetos.** Artigo publicado no site www.apinfo.com: Taubaté, 2004.

[Castells99] Castells, Manuel. **A Era da Informação: Economia, Sociedade e Cultura**. Volume I. A Sociedade em Rede. São Paulo, Paz e Terra, 1999.

[Castro01] Castro, Edison. **Programação e Objeto.** São Paulo: UNIFEO, 2001.

[CTI97] CTI (Fundação Centro Tecnológico para informática). **Qualidade de Software: Visões de Produto e Processo**. Junho, 1997.

[Colombo04] Colombo, R. M. T. **Processo de Avaliação da Qualidade de Pacotes de Software**. Dissertação (Mestrado Profissional na área de Gestão da Qualidade Total) – FEM - Faculdade de Engenharia Mecânica, UNICAMP - Universidade Estadual de Campinas.Campinas, 2004.

[Cordeiro00] Cordeiro, Marco Aurélio. **Métricas de Software**. Artigo publicado no *site* Bate Byte 101. Setembro de 2000.

[Cosmic06] **Overview of the COSMIC-FFP Method**. Artigo extraído do site http://www.cosmicon.com.

[Dumas94] Dumas, J.S. e Redish, J.C. **A Practical Guide to Usability Testing**. Ablex, Norwood, NJ. 1994.

[Furlan98] Furlan, José Davi. Modelagem de Objetos através da UML. Makron Books, 1998.

[Guerra05] Guerra, A. C., Aguayo, M. T. V., Colombo, R. T. **Processo de Avaliação de Produtos de Software** In: Conferência Ibero-Americana WWW/Internet 2005, 2005, Lisboa. International Association for Development of the Information Society. Lisboa: , 2005. p.437 – 442

[Gomes04] Gomes, Álvaro Eduado. **Métricas e Estimativas de Software - O início de um rally de regularidade.** Artigo extraído do site Apinfo.com. São Paulo, 2004.

[GUIA05] GUIA. **Guia de avaliação da Qualidade de Produtos de Software-** http://www.cenpra.gov.br/publicacoes/index.htm. Campinas, 2005

[ISO/IEC 9126], 1991. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ISO/IEC 9126 **Information Technology – Software product evaluation - Quality characteristics and guidelines for their use**;13 p. Geneve ISO Dez / 1991.

[ISO/IEC 9126-1]. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ISO/IEC 9126-1, **Software Engineering - Software product quality - Part 1: Quality Model**; Geneve ISO Jun / 2001.

[ISO/IEC 9126-2]. INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC DTR 9126-2, - **Software Engineering - Software product quality - Part 2: External Metrics.** 2001.

[Kaplan97] Kaplan, Robert; NORTON, David. **A Estratégia em Ação - Balanced Scorecard**. Rio de Janeiro: Campus, 1997.

[Kacuta06] Kacuta, L.Y. **Um Modelo de Negócio para Desenvolvimento de Software** Trabalho Final de Mestrado Profissional apresentado à comissão de Pós Graduação do Instituto de Computação, como requisito para a obtenção do título de Mestre Profissional em Computação, Fevereiro de 2006.

[Kitchenham96] Kitchenham, B. ET AL., **SOFTWARE QUALITY: THE ELUSIVE TARGET**, IEEE SOFTWARE, JANUARY. 1996.

[MCT02] MCT. MINISTÉRIO DE CIÊNCIA E TECNOLOGIA. Brasília: Ministério de Ciência e Tecnologia. Secretaria de Política de Informática. 2002. 258p. (n. 4).

[Macaulay96] Macaulay, Linda A. **Requirements Engeneering**. 1ed. Great Britain: Springer-Verlag London Limited, 1996.

[Machado04] Machado, Marcio P. **Métricas e Qualidade de Software.** Artigo publicado-Universidade Estadual do Espírito Santo. Vitória, 2004.

[Macoratti05] Macoratti, José Carlos. **Estimativas de tamanho de software e APF.** São Paulo: Artigo publicado no site Macorattinet, 2005.

[Martin91] Martin, James. **Engenharia da Informação**: Introdução. Rio de Janeiro: Campus, 1991.

[Oliveira97] Oliveira, A.; RÊGO, C. M.; SOUZA, E. P. de; MARTINEZ, M. R. M.;AGUAYO, M. T. V.; COLOMBO, R. M. T.; MAINTINGUER, S. T. **Método de Avaliação de Qualidade de Produto de Software MEDE-PROS**. FBN número: 135.620, Livro:216 e Folha 84. PI 820166243, 1997.

[Oliveira96] Oliveira, Adelize Generini de. **Análise, Projeto e Programação Orientados a Objetos.** Bookstore, 1996.

[Pressman99] Pressman, R.S. **Engenharia de Software.** São Paulo: Editora Makron, 1999.

[Queiroz00] Queiroz, José E. **Inspeção de Conformidade de Produtos com o Padrão ISO 9241.** Campina Grande, 2000.

[Sanches05] Sanches, Maurício G. **Um Estudo Sobre os Riscos Inerentes à Implantação do Reuso de Componentes no Processo de Desenvolvimento de Software**. Dissertação Mestrado Profissional no Instituto de Computação – IC, UNICAMP - Universidade Estadual de Campinas, 2005

[Schulmeyer98] Schulmeyer, G. Gordon e MCMANUS, James I. **Handbook of Software Quality Assurance**. Editora Prentice Hall PTR, 1998.

[Schwartz75] Schwartz, J. I. ,**Construction of software. In: Practical Strategies for Developing Large Systems. Menlo Park.** Addison-Wesley, 1st. ed., 1975.

[Smith97] Smith, A. G. **Testing the surf: criteria for evaluating internet information sources**. Public-Access Computer Systems Review. 1997

[Sommerville03] Someerville, Ian. Engenharia de Software. 6.ed. São Paulo: Addison Wesley, 2003.

[Simoes99] Simões, Carlos Alberto. **Sistemática de Métricas, Qualidade e Produtividade.** Developers' Magazine, setembro de 1999.

[Yourdon99] Yourdon, EDWARD e ARGILA, CARL. **Análise e Projeto Orientados a Objetos: Estudos de casos**. Traduzido por Angelina Carvalho Gomes e Alvaro Antunes. São Paulo: Makron Books, 1999.

[Schach04]Schach Stepthen R. **Object-Oriented and Classical Software Engineering**. McGraw Hill 2004.