UNIVERSIDADE FEEVALE

JOCEMAR SCHMITT MARIA

ROTEIRO PARA ELABORAR MÉTRICAS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE UTILIZANDO MÉTODOS DMAIC E GQM

Novo Hamburgo
2010

JOCEMAR SCHMITT MARIA

ROTEIRO PARA ELABORAR MÉTRICAS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE UTILIZANDO MÉTODOS DMAIC E GQM

Trabalho de Conclusão de Curso
apresentado como requisito parcial
à obtenção do grau de Bacharel em
Sistemas de Informação pela
Universidade Feevale

Orientador: Roberto Scheid

Novo Hamburgo
2010

AGRADECIMENTOS

Primeiramente a Deus, pela vida e pela força para seguir em frente apesar de todos os obstáculos. Aos meus pais, por me ensinarem os valores da vida e por nunca terem medidos esforços para me proporcionar o conhecimento.

Agradeço em especial minha esposa Clair por ter me apoiado e incentivado na conclusão do curso. Aos nossos filhos Bruna, Júlia e Lucas pela compreensão nos momentos de ausência em favor dos estudos.

A todos os professores, em especial ao meu amigo e orientador Roberto Scheid pela compressão e por ter me ajudado no desenvolvimento deste trabalho.

Enfim a todos que de alguma forma me ajudaram para que este objetivo fosse alcançado.

RESUMO

Mediante as constantes mudanças que ocorrem no setor econômico, as organizações, procuram garantir sua sobrevivência através de inúmeros projetos lançados como tentativa de resposta às pressões e ondas tecnológicas de direções diversas. Normalmente, a quantidade de projetos é diversificada e os investimentos e recursos são significativos. A dificuldade em decidir qual projeto de software priorizar dentre todos, aumenta quando a área de desenvolvimento da empresa não possui métricas para quantificar o esforço para criar-se ou customizar-se um determinado software. Mensurar o esforço necessário no desenvolvimento é possível através de métricas utilizadas na engenharia de software. O problema, basicamente, consiste de 2 (dois) aspectos: 1) conseguir identificar a(s) métrica(s) adequada(s) e/ou; 2) definir a metodologia que melhor se ajusta e que seja possível de ser implantada na organização. Sendo assim, este trabalho tem como objetivo propor um modelo (roteiro) que possa ajudar na definição de métricas há serem aplicadas no processo de desenvolvimento de software. Propõe-se que tais métricas sejam oriundas da junção dos métodos GQM e melhoria contínua (DMAIC).

Palavras-chave: Métricas, Desenvolvimento de Software, Melhoria contínua, GQM, DMAIC.

ABSTRACT

Through the constant changes that occur in the economic sector, the organizations, seek to ensure their survival through numerous projects launched in an attempt to respond to pressures and wave technology of different directions. Typically, the amount of projects is diverse and investment and resources are significant. The difficulty in deciding which project to prioritize among all software, increases when the area's development company does not have metrics to quantify the effort to create it or to customize a particular software. Measure the effort needed in the development of metrics, which are used in software engineering. The problem basically consists of 2 (two) things: 1) can identify the person (s) metric (s) right (s) and / or; 2) establish a methodology that best setting and be able to be deployed in the organization. Thus, this paper aims to propose a model (script) that can help define metrics is being applied in the process of software development. It is proposed that such metrics are derived from the junction of the GQM method and continuous improvement (DMAIC).

Key words: Metrics, Software Development, Continuous Improvement, GQM, DMAIC.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

APF	Análise de pontos de função
COCOMO	*Constructive Cost Model*
DMAIC	*Define Measure Analyze Improve Control*
FP	*Function Point*
GCO	Gerência de Configuração
GQA	Garantia da Qualidade
GQM	*Goal Question Metric*
IBM	*International Bussines Machine*
IEC	*International Electronics Community*
IEEE	*Instituto de Engenharia Elétrica e Eletrônica*
IFPUG	*International Function Point User Group*
ISO	*International Organization for Standardization*
KDSI	Milhares de Instruções-Fonte Disponibilizadas
KLOC	Mil Linhas de Código
LOC	Linhas de Código
MED	Medição
MPS.BR	Melhorias de Processo do Software Brasileiro
SEI	*Software Engineering Institute*
SIX SIGMA	Seis Sigma
SQuaRE	*Software product Quality Requirements and Evolution*
TI	Tecnologia da Informação

SUMÁRIO

INTRODUÇÃO

Quando do desenvolvimento de um novo software, o esforço efetuado em melhorias – e/ou correções de erros encontrados nas suas funcionalidades – ocasionam dificuldades quando se trata de dimensionar o esforço que será necessário para construí-lo e, consequentemente, o custo desta obra. O fato do software "não ser algo tangível" na maioria das vezes torna difícil de entender o porquê do mesmo de levar determinado tempo para ficar pronto. Diferente de construir um prédio, em que o esforço empregado pode ser visualmente percebido. Neste sentido, as métricas utilizadas para medir o que se constrói são de extrema importância. Segundo DeMarco (1991, p.51), a métrica é o número que representa uma idéia, um índice quantitativo, algo mensurável. Mas como quantificar um software? Pressman (1995, p.75) na década de 90 enfatizou que a maior parte dos desenvolvedores de software não realizava medições. Além disso, o processo de implantação de uma metodologia é difícil devido o fato da maioria não ter muita vontade de começar a medir. Não obstante, iniciar um novo processo, compilar medidas quando no passado não era feito, frequentemente, provoca resistências. Pressman (1995, p.75) complementa ainda que os benefícios que se obtêm com a medição de software são tão motivadores que o trabalho árduo vale o esforço de implementar uma metodologia.

De acordo com Pressman (1995, p.4), durante as três primeiras décadas da era do computador o foco da indústria da computação era reduzir o custo do processamento e armazenamento de dados. Ao passar do tempo, durante a década de 80, avanços da tecnologia possibilitaram ter um maior processamento a um custo cada vez mais baixo. Durante a década de 90 a realidade é diferente. O foco está em melhorar a qualidade das soluções fornecidas através de um "programa" de computador.

Pressman (1995, p.9) também enfatiza que o software, ao contrário do hardware, frequentemente é o item de maior custo. A preocupação em saber porque demora tanto tempo para que os programas sejam concluídos, leva os gerentes a refletir e fazer as seguintes perguntas:

- Por que os custos são tão elevados?
- Por que não é descoberto todos os erros antes de entregar o software ao cliente?
- Por que é difícil medir o progresso enquanto o software é desenvolvido?

Estas questões mostram a preocupação relativa ao software e a maneira pela qual é desenvolvido. Preocupação esta que levou ao desenvolvimento e adoção de práticas de engenharia de software. Procurando melhorar os processos de desenvolvimento, surge a medição de software como forma de não efetuar o dimensionamento baseado no *feeling* e/ou na experiência do analista de sistemas.

> Não dedicamos tempo para coletar dados sobre o processo de desenvolvimento de software. Com poucos dados históricos como guia, as estimativas tem sido "a olho", com resultados previsivelmente ruins. Sem nehuma indicação sólida de produtividade, não podemos avaliar com precisão a eficácia de novas ferramentas, métodos ou padrões (PRESSMAN, 2007, p.23).

Uma das justificativas de utilizar medição de software é a possibilidade de acompanhar se o processo está sendo melhorado ou não, pois se não houver medidas anteriores não será possível efetuar as comparações, e com isso estabelecer metas de melhorias dos processos.

Para Sommerville (2003, p.438-439), 2 (dois) tipos de medidas são utilizadas:

- Medidas relacionadas ao tamanho de determinada saída de uma atividade. Sendo que, destas, a mais comum é o número de linhas de código-fonte;
- Medidas relacionadas a contagem de funções, onde são observadas as funcionalidades do software entregue. A melhor medida conhecida deste tipo é a contagem por pontos de função.

Paula Filho (2005, p.239-240) salienta que técnicas que estimam tamanho de software devem atender aos seguintes critérios:

- ser contável através de um procedimento bem definido;
- ser calculável a partir da especificação dos requisitos de software;
- apresentar boa correlação com o esforço de desenvolvimento.

Para conseguir um melhor entendimento sobre medição e métricas de qualidade, buscou-se informações em bibliografias que tratam a respeito do assunto. A partir disso, foi possível sugerir um modelo com o objetivo de apoiar a definição de métricas.

O objetivo geral do presente estudo consiste num modelo (roteiro) que possa ajudar na definição de métricas há serem aplicadas no processo de desenvolvimento de software. Propõe-se que essas métricas sejam oriundas da junção dos métodos GQM e melhoria contínua.

Os objetivos específicos foram assim definidos: 1) efetuar pesquisa bibliográfica sobre métricas e medições; 2) analisar fundamentação teórica sobre técnicas de medição de software mais utilizadas; 3) Definir um modelo (roteiro) para apurar métricas; e 4) validar o roteiro proposto, aplicando-o numa empresa (estudo de caso).

O trabalho está dividido em 3 (três) capítulos, os quais são compostos por seções. No primeiro capítulo será abordado a importância, a evolução e as características do software. Além de como é o processo de desenvolvimento de software e definição para medidas, métricas e indicadores. Ainda neste capítulo, um estudo da bibliografia de métricas de qualidade sugeridas pela ISO/IEC 9126 e o modelo proposto para definir métricas utilizando GQM e DMAIC.

O segundo capítulo, descreve a metodologia utilizada (pesquisa-ação), o método de pesquisa escolhido para obter dados a fim de validar a proposta elaborada.

No terceiro capítulo, apresenta-se a implementação do plano de análise dos dados. De modo a validar o modelo proposto foi utilizado análise de conteúdo.

Finalizando o trabalho, encontra-se a conclusão, e sugestões para trabalhos futuros.

1 FUNDAMENTAÇÃO TEÓRICA

O referencial teórico que será apresentado tem como objetivo oferecer embasamento para o atendimento dos objetivos propostos na pesquisa.

1.1 Software

Segundo Pressman (2002, p.3), software é o produto que os engenheiros de software projetam e constroem. Abrangem programas que executam em computadores de qualquer tamanho e arquitetura. O software afeta praticamente todos os aspectos de nossas vidas e tornou-se difundido no nosso comércio, na nossa cultura e nas nossas atividades do dia-a-dia.

1.1.1 Importância

O software foi adquirindo importância, com a melhora no desempenho do hardware e com as profundas modificações na arquitetura dos computadores, onde é indispensável ter sistemas mais sofisticados e complexos. Esta relevância trouxe também a necessidade de se pensar que o software deve ser um produto de qualidade e que – ao mesmo – precisa transmitir confiança, pois faz parte do dia-a-dia empresarial (PRESMANN, 2002, p.4).

Pressman (2002, p.4) afirma ainda que o software entrega o mais importante produto da nossa época: a informação.

> O software transforma dados pessoais (p. ex., as transações financeiras de uma pessoa) de modo que possam ser mais úteis em determinado contexto; gera informação comercial para melhorar a competitividade; fornece um portal para as redes de informação de âmbito mundial (p. ex., a Internet) e proporciona os meios para obter informação em todas suas formas (PRESSMAN, 2002. p.4).

Vários benefícios são contemplados pela implementação de métricas, alguns são citados por Pressman (1995, p.60): "... 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".

1.1.2 Evolução

Ao passar do tempo, desde o início do computador, o software foi ganhando espaço no mundo da tecnologia. O software é tanto o produto e, ao mesmo tempo, o veículo para a entrega do produto (PRESSMAN, 2002, p.4). Como produto, ele é o transformador de informação, quer resida em um computador de grande porte ou em um celular, produzindo,

gerando, modificando ou transmitindo informação. Seja do tamanho que for, pode ser tão simples como um único bit ou uma complexa apresentação em multimídia.

Com a evolução do software surgiu também a necessidade de melhorar a qualidade do desenvolvimento do software. As equipes de especialistas de softwares passaram a responder pela produtividade, custos de desenvolvimento, quanto a qualidade do produto entregue – e também – quanto a dificuldade de visualizar o produto enquanto este é desenvolvido. Estas questões são manifestações da preocupação sobre o software e a maneira como ele é desenvolvido – preocupação que leva a adoção da prática de engenharia de software (PRESSMAN, 2002, p.6).

Para Peters (2001, p.35), o papel do *feedback* no processo de software, possui um "efeito-onda", onde as informações relativas às falhas e às mudanças de requisitos são retornadas ao estágio inicial do ciclo. As respostas ao feedback fazem com que os produtos evoluam.

1.1.3 Característica

O software é o elemento de um sistema lógico e o hardware de um sistema físico. Apresenta características que o diferenciam do hardware apesar de ambos dependerem de pessoas. Sendo estas descritas abaixo (PRESSMAN, 1995, p.13-16):

- É desenvolvido ou passa por um processo de engenharia, não sendo manufaturado no sentido clássico;
- Não "se desgasta";
- Apesar da indústria estar se movendo em direção a montagem baseada em componentes, a maior parte de software continua a ser construída sob encomenda.

1.2 Processo de desenvolvimento

O desenvolvimento de software envolve uma sequência de atividades, essas atividades englobam o processo de software (PETERS, 2001, p.29). Pressman (2002, p.17) vai mais profundo na preocupação com a qualidade e prazos, definindo processo de software como sendo uma série de passos previsíveis – um roteiro que o ajuda a criar a tempo um resultado de alta qualidade. Paula Filho (2003, p.11) define processo como sendo um conjunto de passos parcialmente ordenados, constituídos por atividades, métodos, práticas, e transformações, usadas para atingir uma meta. Um processo é definido quando tem

documentação que detalha: O que é feito? Quando? Por quem? As coisas que usa e as coisas que produz. Os passos de um processo podem ter ordenação apenas parcial, o que permite executar passos em paralelo (figura 1).

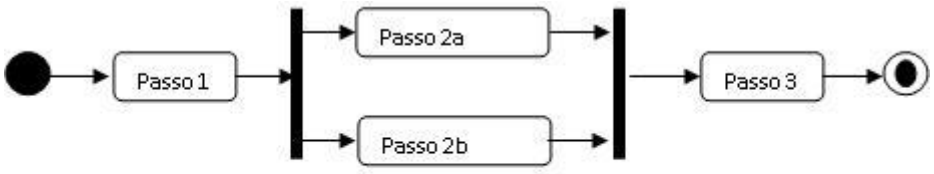


Figura 1 - Passos de um processo
Fonte: Paula Filho (2003, p.11).

Segundo Peters (2001, p.39), durante a fase de desenvolvimento de software há 3 (três) processos principais identificados no padrão IEEE 1074-1995:

a) ***requisitos*** – decidir o que o sistema deve fazer, suas atividades, riscos e plano de testes;

b) ***projeto*** – determinar como um sistema efetua os cálculos, sua estrutura e suas funções específicas;

c) ***implementação*** – produzir código-fonte, documentação e testes; validar e verificar.

Paula Filho (2003, p.4) caracteriza o ciclo de vida do software conforme figura 2.

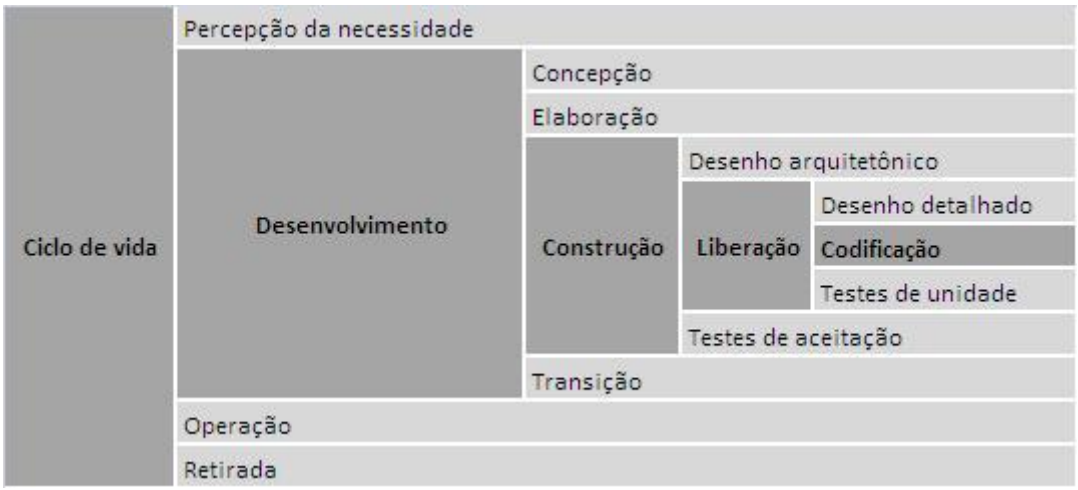


Figura 2 - Esquema simplificado do ciclo de vida do software
Fonte: Paula Filho (2003, p.4).

Ainda de acordo com o autor, como todo produto industrial, o software tem um ciclo de vida conforme descrito a seguir:

- Ele é concebido a partir da percepção de uma necessidade;
- É desenvolvido, transformando-se em um conjunto de itens entregue a um cliente;
- Entra em operação, sendo usado dentro de algum processo de negócio e sujeito a atividades de manutenção, quando necessário;
- É retirado de operação ao final de sua vida útil.

É interessante observar na figura 2, que a etapa de codificação – que representa a escrita final de um programa na forma inteligível pelo computador – é apenas uma pequena parte do ciclo de vida.

1.3 Medidas, métricas e indicadores de software

Peters (2001, p.431) conceitua medição de software como sendo uma técnica ou um método que aplica as medidas de software a uma classe de objetos de engenharia de software de forma a atingir um objetivo predefinido.

A gestão quantificada de um processo de software envolve medições tanto de risco como de desempenho da equipe. Conhecer é importante. Quantificar este conhecimento envolve introduzir medições (números) no processo. As medições fornecem um *feedback* valioso para os planejadores, ajudam no controle e na melhoria do processo, pois quantificam a avaliação de um produto de software (PETERS, 2001, p.430).

Corroborando com a idéia de Peters, Pressman (2002, p.75) escreve que a medição permite obter entendimento visto que possibilita um mecanismo para uma avaliação objetiva.

> Uma medida de software é o mapeamento de um conjunto de objetos no mundo da engenharia de software em um conjunto de construções matemáticas, tais como números ou vetores de números (McCLURE apud PETERS, 2001, p.431).

Peters diz ainda que na área da engenharia de software é comum a utilização do termo métrica do software. Essa métrica é uma medição quantitativa simples derivada de qualquer atributo do ciclo de vida do software. Utilizações comuns do termo métrica na engenharia de software foram identificadas por Fenton (apud PETERS, 2001, p.430) e são resumidos no quadro a seguir.

Quadro 1 - Formas de métricas de software

Interpretação da "métrica do software"	Exemplo
Número derivado de um processo de software, produto ou recurso	Linhas de Código (LOC) por programador – mês
Escala de medição	Classificação nominal das falhas do software
Atributo de software identificável	Independência de plataforma de um programa
Modelo direcionado a dados (descreve uma variável dependente que é uma função de variáveis independentes)	Esforço como uma função de milhares de instruções-fonte fornecidas (KDSI) no modelo *Constructive Cost Model* (COCOMO) de Boehm (Boehm, 1981)

Fonte: (PETERS, 2001, p.430).

Existe, atualmente, um número significativo de medidas de software que são utilizados. No entanto, não há um consenso sobre quais dessas medidas são mais significativas e úteis. Peters (2001, p.432) cita que as medidas utilizadas com mais frequência são as medidas de tamanho e medida de linha de código.

A maioria das empresas de desenvolvimento de software costumam ter menos de 20 (vinte) pessoas na área. Diante disso, não se pode esperar que estas empresas desenvolverão programas abrangentes de métricas de software. Mas, é razoável sugerir que estas empresas meçam e depois façam uso destas medidas para aperfeiçoar o seu processo de software (PRESSMAN, 2002, p.100).

Kautz apud Pressman (2002, p.99-100) sugere uma abordagem de bom senso para a implementação de qualquer atividade relacionada a processo de software. Tornar simples, ajustar as necessidades locais, certificando-se que agrega valor as organizações são diretrizes sugeridas por Kautz.

Durante o período de aprendizado, o custo de coletar medidas e computar métricas, para pequenos grupos, varia de 3 a 8% do orçamento do projeto. Depois que houver a familiarização do processo pelos engenheiros de software e gerentes de projeto este percentual cai para 1% (GRABLE apud PRESSMAN, 2002, p.101).

Ao abordar métricas durante o processo de desenvolvimento de software é importante conhecer as diferenças entre os conceitos de medida, métrica e indicador. O quadro a seguir esclarece o conceito de cada item.

Quadro 2 - Definições de medida, métrica e indicador

ITEM	DEFINIÇÃO
Medida	De acordo com o *Instituto de Engenharia Elétrica e Eletrônica* (IEEE), uma medida é uma avaliação em relação a um padrão. Conforme McGarry apud Sato (2007, p.40), uma medida é a avaliação de um atributo segundo um método de medição específico, funcionalmente independente de todas as outras medidas e capturando informação sobre um único atributo. Um exemplo de medida é 5 cm. Centímetro é o padrão e 5 é a medida, que indica quantos múltiplos ou frações do padrão estão sendo representados. Em desenvolvimento de software, um exemplo de medida pode ser o número de linhas de código. Segundo Sato (2007, p.40), "No entanto, não existe um padrão universal para representar linhas de código, pois as linguagens podem variar, assim como as regras para cálculo de linhas de código. Portanto, uma medida pode ser baseada em um padrão local ou universal, mas o padrão precisa ser bem definido". O MPS.BR (SOFTEX, 2009) complementa ainda que medida é uma variável para a qual o valor é atribuído como um resultado de uma medição, que podem ser classificadas em: básicas ou derivadas. Medida básica quando é definido em termos de um único atributo por método de medição, não dependendo de outras medidas (por exemplo: peso, altura, LOC). Medida derivada é aquela definida em função de dois um mais valores de medidas básicas ou derivadas (por exemplo: produtividade, que é o resultado de LOC / por horas trabalhadas). Vazquez (2008, p.31) explica medida como sendo a quantificação de uma característica. Complementa ainda que no caso da área de sistemas devem ser avaliadas não só suas características de produto final, mas também as características dos processos envolvidos em sua concepção. No artigo intitulado *"Software modeling and measurent: The Goal/Question/Metric paradim"* Basili apud Vazquez (2008, p.31) apresentou um modelo bastante ilustrativo e útil no processo de identificação dessas características: para cada um dos objetivos que se deseja acompanhar é possível estabelecer um conjunto de perguntas que verifique o seu cumprimento, para muitas dessas perguntas é possível identificar uma métrica que possa quantificar a resposta.
Métrica	Uma métrica é um método para determinar se um sistema, componente ou processo possui certo atributo (IEEE apud SATO, 2007, p.40). Ela é geralmente calculada ou composta por duas ou mais medidas. Um exemplo de métrica é o número de defeitos encontrados após a implantação: as medidas que compõem essa métrica são o número de defeitos e a fase (ou data) onde o defeito foi identificado (SATO, 2007, p.40). Para Pressman (2002, p.77), métrica de software relaciona as medidas individuais de alguma forma (por exemplo: um número médio de erros encontrado por revisão ou um número médio de erros encontrados por pessoa-hora, empregada nas revisões).
Indicador	Um indicador é um dispositivo ou variável que pode ser configurado para um determinado estado com base no resultado de um processo ou ocorrência de uma determinada condição. Por exemplo: um semáforo ou uma "flag". Pela definição do IEEE, um indicador é algo que chama a atenção para uma situação particular. Ele normalmente está relacionado a uma métrica e provê a interpretação daquela métrica numa determinada situação ou contexto. Sempre que alguém interpreta alguma métrica, está considerando algum tipo de indicador, seja ele algum valor base ou outra métrica. Por exemplo: um aumento substancial no número de defeitos encontrados na última versão pode ser um indicador de que a qualidade do software piorou (SATO, 2007, p.41). MPS.BR (SOFTEX, 2009) define indicador como sendo uma estimativa ou avaliação que provê uma base para a tomada de decisão, podendo ser obtido a partir de medida básica ou derivada. É – via de regra – representado e comunicado por meio de tabelas ou gráficos (por exemplo: de linha, de barra, de dispersão) e possui uma explicação de como os interessados podem interpretar seus resultados, bem como utilizá-los para a tomada de decisão.

Ampliando o que Sommerville (2003, p.438-439) diz a respeito dos tipos de medidas utilizadas, os subitem 1.3.1 e 1.3.2 explicam de forma mais completa cada tipo de medida.

1.3.1 Medidas orientadas a tamanho

Pressman (2002, p.84) ressalta que métricas orientadas a tamanho são originadas pela normalização das medidas de qualidade e/ou produtividade. Ao manter registros simples é possível criar uma tabela para cada projeto conforme o quadro 3.

Quadro 3 - Métricas orientadas a tamanho

Projeto	LOC	Esforço	$(000)	Pág.doc.	Erros	Defeitos	Pessoas
alfa	12.100	24	168	365	134	29	3
beta	27.200	62	440	1.224	321	86	5
gama	20.200	43	314	1.050	256	64	6
-	-	-	-	-	-		
-	-	-	-	-	-		
-	-	-	-	-	-		

Fonte: (PETERS, 2002, p.84).

A partir dos dados contidos no quadro 3, um conjunto de métricas simples orientadas a tamanho pode ser desenvolvido:

- erros por KLOC;
- defeitos por KLOC;
- $ por LOC;
- páginas de documentação por KLOC.

Métricas orientadas a tamanho não são universalmente aceitas como sendo o melhor modo de medir o processo de desenvolvimento de software (JONES apud PRESSMAN, 2002, p.63). Os que apóiam a medida argumentam que LOC é um “artefato” de todos os projetos de desenvolvimento de software, que pode ser facilmente contado. Por outro lado, os oponentes argumentam que as medidas de LOC são:

- dependentes da linguagem de programação;
- penalizam programas bem projetados, porém mais curtos;
- podem acomodar facilmente linguagens não-procedimentais.

1.3.2 Medidas orientadas a função

Pressman (2002, p.84) explica que métricas orientadas a função usam uma medida da funcionalidade entregue pela aplicação. Como "funcionalidade" não pode ser medida diretamente, deve ser originada indiretamente utilizando outras medidas diretas. A técnica da análise de pontos de função surgiu na IBM[1], no início da década de 70, como uma alternativa às métricas baseadas em linhas de códigos. Na época, Allan Albrecht foi encarregado de medir a produtividade de vários projetos de software desenvolvidos pela IBM. Estes projetos tinham sidos desenvolvidos em uma grande variedade de linguagens de programação. Isso motivou a busca por uma medida que fosse independente da linguagem de programação utilizada (VAZQUEZ, 2008, p.36). A partir do estudo realizado, a IBM propôs a análise de pontos de função (APF). Pontos de função são calculados completando a tabela 1.

Tabela 1 - Cálculo dos pontos por função

			Fator de peso			
Parâmetro de medição	Contagem		Simples	Médio	Complexo	
Quantidade de entradas do usuário		x	3	4	6	
Quantidade de saídas do usuário		x	4	5	7	
Quantidade de consultas do usuário		x	3	4	6	
Número de arquivos		x	7	10	15	
Quantidade de interfaces externas		x	5	7	10	
Contagem total						

Fonte: (PRESSMAN, 2002, p.84).

Pressman (2002, p.85-86) relaciona 5 (cinco) características do domínio da informação que são definidas, após são contadas e registradas nos lugares próprios da tabela. Os valores do domínio da informação são definidos da seguinte maneira:

a) ***quantidade de entradas do usuário (EE[2])*** – cada entrada do usuário, que fornece dados distintos orientados à aplicação do software, é contada;

b) ***quantidade de saídas do usuário (SE[3])*** – cada saída do usuário, que fornece informação orientada à aplicação para o usuário é contada. Neste contexto, saída refere-se a relatórios, telas, mensagens de erros, etc;

[1] International Bussines Machine.
[2] Entrada externa, processa dados ou informações de controle recebido de fora da fronteira da aplicação (VAZQUEZ, 2008, p.101)
[3] Saída externa, que envia dados ou informações de controle para fora da fronteira da aplicação (VAZQUEZ, 2008, p.101).

c) ***número de consultas do usuário (CE[4])*** – uma consulta é definida como uma entrada on-line, que resulta na geração de alguma resposta imediata do software sob forma de uma saída on-line. Cada consulta distinta é contada;

d) ***quantidade de arquivos (ALI[5])*** – cada arquivo mestre lógico[6] é contado;

e) ***quantidade de interfaces externa (AIE[7])*** – todas as interfaces em linguagem de máquina, que são usadas para transmitir informação a outro sistema, são contadas.

Uma vez coletados esses dados, um valor de complexidade é associado com cada contagem.

Depois de calculados, os pontos por função são usados de modo análogo à LOC como forma de normalizar medidas de produtividade, qualidade e outros atributos de software:

- erros por FP;
- defeitos por FP;
- $ por FP;
- páginas de documentação por FP;
- FP por pessoa-mês.

As métricas devem resultar em um valor matemático que verifique se o produto de software tem qualidade (SODRÉ, 2006, p.34).

1.4 Características de qualidade observadas na escolha de uma métrica

Conforme Koscianski (2007, p.225), todo o trabalho de avaliação é colocado em risco se não for possível garantir uma coleta dos dados confiáveis. Quando um resultado é utilizado para julgar um software é preciso ter certeza que ela está sendo feito de maneira

[4] Consulta externa, que envia dados ou informações de controle para fora da fronteira da aplicação (VAZQUEZ, 2008, p.102).
[5] Arquivo lógico interno, um grupo de dados ou informações de controle mantido pela aplicação (VAZQUEZ, 2008, p.73).
[6] O termo *arquivo* não significa um arquivo do sistema operacional, como é comum na área de processamento de dados. Ele se refere a um grupo de dados logicamente relacionados e reconhecido pelo usuário (VAZQUEZ, 2008, p.72).
[7] Arquivo de interface externa, um grupo de dados ou informações de controle que não é mantido pela aplicação (VAZQUEZ, 2008, p.74).

correta. Além da precisão dos resultados, fatores como o custo também podem influir na escolha das métricas a serem utilizadas para avaliar um produto.

Segundo o modelo SQuaRE, um avaliador ao elaborar novas métricas, ou escolher uma delas a partir de uma biblioteca pode considerar alguns critérios listados a seguir (SCALET apud KOSCIANSKI, 2007, p.226):

- ***significância*** – os resultados da medição devem agregar informação útil sobre a qualidade, métricas consideradas de pouca importância deveriam ser descartadas;
- ***custo e complexidade*** – a aplicação da métrica deve ser viável tanto economicamente como tecnicamente. Por exemplo, uma métrica pode não ser aplicável por falta de laboratórios ou porque requer o auxílio de um especialista;
- ***repetibilidade*** – o resultado de uma medida deve ser o mesmo quando aplicado em um mesmo produto, com a mesma especificação de avaliação, com o mesmo ambiente, mesmos avaliadores e usuários-teste;
- ***reproducibilidade*** – este critério é mais rigoroso que o anterior, pois significa que os resultados de uma medição deve ser o mesmo para avaliadores diferentes;
- ***validade*** – a métrica deve permitir que sua corretude, precisão ou margem de erro dos resultados da avaliação sejam demonstradas. É importante salientar que: características de software e de hardware; avaliadores e modelos matemáticos podem influenciar os resultados;
- ***objetividade*** – os resultados da medição não podem ser influenciados por subjetividade do avaliador ou outros agentes que realizem a avaliação, tais como usuários de teste. Quando a métrica necessitar essa subjetividade, algum mecanismo de ajuste deve ser adotado;
- ***imparcialidade*** – o processo de medição não deve ser tendencioso. Um exemplo de aplicação tendenciosa seria avaliar dois software num mesmo ambiente sendo que um deles tem como requisito a utilização desse ambiente, enquanto o outro não.

Por existirem muitas medidas diferentes para software, é importante saber escolher quais serão mais adequadas a um projeto. Esta escolha deve analisar características como o custo de sua aplicação e, o que é obvio, considerar os objetivos do projeto (KOSCIANSKI, 2007, p.224).

1.4.1 Escolha de métricas a partir dos objetivos

O *Software Engineering Institute* (SEI) desenvolveu um livro de diretrizes abrangente (PARK apud PRESSMAN, 2002, p.101) para estabelecer um programa de métricas de software "orientado a metas". O livro de diretrizes sugere os seguintes passos:

- identifique suas metas de negócio;
- identifique o que você deseja saber ou aprender;
- identifique suas submetas;
- identifique as entidades e atributos relacionadas a suas submetas;
- formalize suas metas de medição;
- identifique questões quantificáveis e os indicadores correlacionados que você vai usar para ajudá-lo a atingir suas metas de medição;
- identifique os elementos de dados que você vai coletar para construir os indicadores que ajudam a responder suas perguntas;
- defina as medidas a serem usadas e torne essas definições operacionais;
- identifique as ações que você irá executar para implementar as medidas;
- prepare um plano para implementar as medidas.
- para Koscianski (2007, p.224), uma maneira organizada de tratar o planejamento da medição é utilizando o método GQM (*Goal-Question-Metric*), o princípio enunciado por Basili apud Koscianski é simples:

> "Especificar objetivos é por si próprio um tópico importante, pois, sem objetivos, há o risco de coleta de dados sem relação e sem importância" (KOSCIANSKI, 2007, p.224).

1.4.2 Método GQM

O GQM é um método que ajuda a definir e integrar objetivos a modelos de processo, produto e perspectivas de qualidade baseada em necessidades específicas do projeto e organizações através de um programa de medições. Ou seja, alinha as medições necessárias aos projetos de softwares com objetivos e metas da organização. A figura 3 representa a forma como o método é estruturado (SOFTEX, 2009).

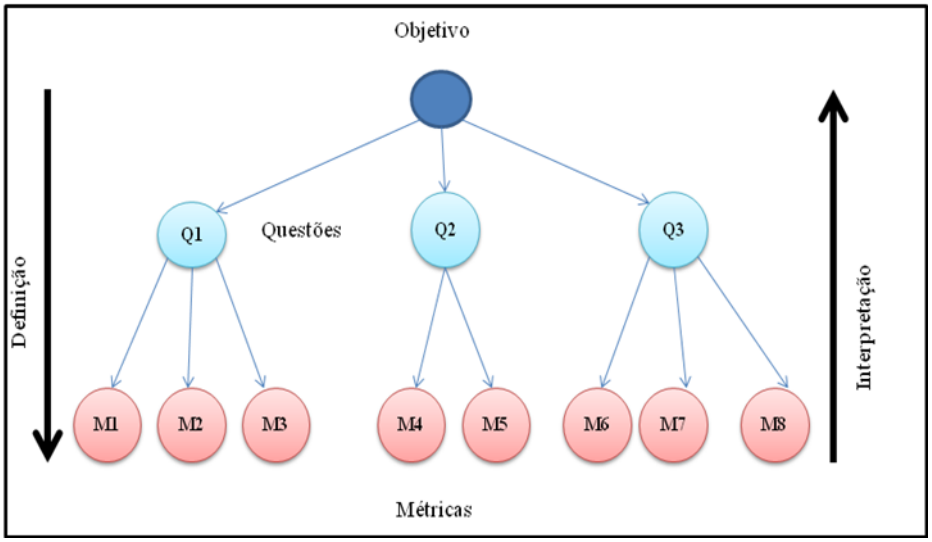


Figura 3 - A estrutura do GQM
Fonte: Solingen apud Covatti (2007, p.55).

A figura 4 apresenta o modelo hierárquico da abordagem GQM, o qual se inicia no nível superior com a definição dos objetivos de medição. Na sequência, estes objetivos são refinados em diversas questões. E por fim, de cada questão são extraídas métricas que devem prover informações para responder as questões levantadas. As questões são perguntas e discussões da equipe de acordo com cada um dos objetivos definidos pela direção (AIZAWA, 2007, p.31).

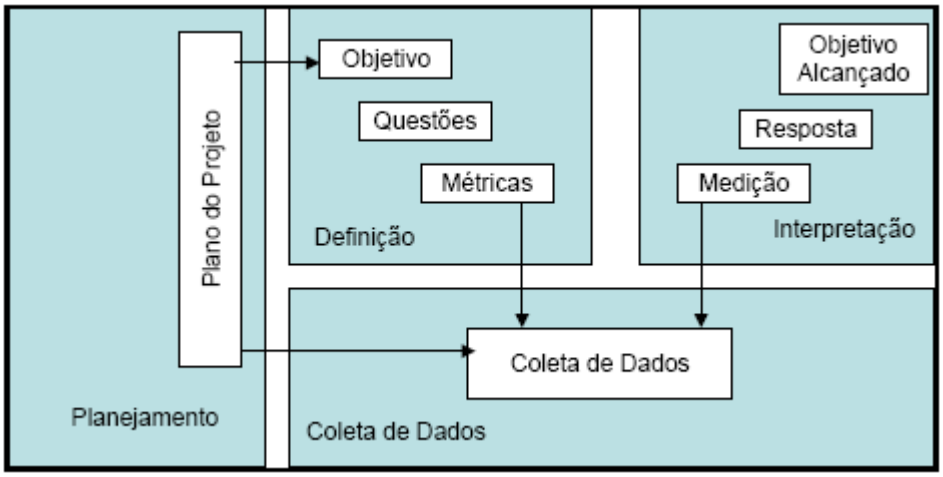


Figura 4 - As fases do método GQM
Fonte: Solingen apud Covatti (2007, p.55).

O rastreamento mantido entre objetivos e métricas, serve como base para futuras adaptações e melhorias do processo de medição. Quando os objetivos mudam, o reflexo no conjunto de métricas pode ser facilmente percebido, e então elas podem ser adaptadas às novas necessidades (AIZAWA, 2007, p.31).

Esse mesmo rastreamento se percorrido no sentido oposto a partir do nível inferior, pode ser usado como um guia para a interpretação do resultado das medições. Os dados coletados são usados para responder às perguntas, e – então – é possível conhecer a situação atual da organização em relação aos objetivos estabelecidos (BORGES apud AIZAWA, 2007, p.31).

De acordo com a figura 4, o método GQM é composto por 4 (quatro) fases:

a) planejamento;

b) definição;

c) coleta de dados;

d) interpretação.

A seguir, descreve-se cada fase que faz parte do método GQM (SOLINGEN apud COVATTI, 2007, p.55-56).

- ***Planejamento*** – Nessa fase, será feito o planejamento para estabelecer um programa de medição, que se dá coletando todas as informações necessárias para iniciar o processo, preparando e motivando as pessoas para a implantação do processo de medição. O plano do projeto contém a documentação dos procedimentos, cronogramas e objetivos do programa de medição. A execução dessa fase completa os requisitos para que o programa de métricas tenha êxito. A fase de planejamento pode ser dividida em 4 (quatro) subfases para melhor distribuição do trabalho:

  a) definição do time;

  b) seleção da área de melhoria;

  c) selecionar projeto de aplicação;

  d) plano do projeto.

- ***Definição*** – A principal tarefa, nessa fase, é decidir quais serão as medidas, incluindo as definições das questões e hipóteses a serem comprovados, revisões, coleta, medições e planos de análise. A fase de definição pode ser dividida em 9 (nove) subfases:

  a) definição dos objetivos de medição;

  b) modelo de processo de software;

c) entrevistas GQM;

d) questões e hipóteses;

e) métricas;

f) plano GQM;

g) plano de medição;

h) plano de análise;

i) revisão.

- ***Coleta de Dados*** – Após todas as atividades de definição terem terminado, a medição pode começar. O sucesso agora depende da acuracidade das métricas coletadas. Na maioria das vezes, os dados podem ser coletados sem intervenção humana, ou seja, não precisam ser agrupados e digitados manualmente. Quando a coleta automática não é possível, um amplo esforço se fará necessário assim também como disciplina na execução dos procedimentos de medição. Os dados coletados são armazenados para serem analisados e essa fase de coleta de dados pode ser dividida em outras 6 (seis) subfases, das quais se destacam:

  a) medição do piloto;

  b) base de métricas;

  c) formulários de coleção de dados;

  d) armazenamento de dados mensurados;

  e) análise;

  f) apresentação.

- ***Interpretação*** – Essa é a fase final e essencial do método GQM. É nesta etapa que os dados coletados são utilizados para responder questões e identificar se os objetivos foram atingidos. Em outras palavras, se as conclusões e as hipóteses são consistentes e positivas para garantir o sucesso da medição. As principais tarefas executadas nessa fase são:

  a) sessões de retro-alimentação;

  b) resultados das medições;

  c) análise de custo e beneficio do programa.

A implementação dos passos sugeridos pelo SEI em um programa de implantação de métricas, além da utilização do método GQM, contribuirá para que a empresa possa alcançar a implantação do processo de certificação MPSBR nível F.

1.4.3 Implementação nível F (MPS.BR)

O guia de implementação MPS.BR fornece orientações para implementar nas empresas os níveis de maturidade descritos no Modelos de Referência MR-MPS. Dentre os níveis de maturidade encontra-se o nível F. O mesmo foca em agregar processos de apoio à gestão do projeto no que diz respeito a Garantia da Qualidade (GQA) e Medição (MED), bem como àqueles referentes à organização dos artefatos de trabalho por meio da Gerência de Configuração (GCO). O propósito do processo de medição segundo a ISO/IEC 12207 (ISO/IEC, 2008) é coletar, armazenar, analisar e relatar os dados relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos, de forma a apoiar os objetivos organizacionais.

A necessidade de se medir em engenharia de software pode ser resumida em uma frase conhecida de DeMarco: “Não se pode controlar o que não se pode medir” (DEMARCO apud SOFTEX, 2009, p.38). Uma boa gestão do processo de software supõe a possibilidade de predizer o comportamento futuro dos produtos e processos de software. Nesses casos, a medição torna-se importante, uma vez que “não se pode dizer o que não se pode medir” (FENTON; PFLEEGER Apud SOFTEX, 2009, p.38). É um dos principais processos para gerenciar as atividades do ciclo de vida e avaliar a viabilidade dos planos de projeto.

A medição tem como principal foco apoiar a tomada de decisão em relação aos projetos, processos e atendimento aos objetivos organizacionais. No nível F, as medições são criadas de forma organizada a partir dos objetivos organizacionais e das necessidades estratégicas de informação da organização.

Com a implementação do nível F (MPS.BR), espera-se alcançar os resultados que seguem:

- objetivos de medições estabelecidos e mantidos a partir dos objetivos de negócio da organização e das necessidades de informação de processos técnicos e gerenciais;

- um conjunto adequado de medidas, orientado pelos objetivos de medição, identificado e definido, priorizado, documentado, revisado e, quando pertinente, atualizado;
- procedimentos especificados para a coleta e o armazenamento de medidas;
- procedimentos especificados para a análise das medidas armazenadas;
- resultado das análises quanto aos dados armazenados;
- tomadas de decisões com base nos resultados das análises efetuadas.

Um processo de medição - em geral - é implementado de forma evolutiva dentro da organização, pois é consequência da maturidade de outros processos. Para chegar aos resultados esperados é importante implementar um método de melhoria contínua.

1.5 Melhoria contínua com DMAIC (SIX SIGMA)

Empresas desenvolvedoras de software têm almejado a excelência na qualidade de seus produtos em decorrência da crescente exigência de seus clientes. Isso faz com que seja necessário a busca por métodos que possam auxiliar a organização alcançar este objetivo. métodos como o DMAIC são responsáveis por possibilitar o constante acompanhamento da evolução de seus processos.

Muitas empresas apesar de terem ouvido relatos sobre experiências de sucesso com a implantação do método DMAIC (SIX SIGMA), tem receio de implantar este programa em suas empresas de software. Entendendo que desenvolver sistemas é diferente da indústria de manufatura, assim como os seus processos são diferentes dos processos industriais. Não conseguindo – por conseguinte – ver que o Six Sigma é uma metodologia para implantação de melhoria contínua possível de ser aplicada em qualquer processo (COVATTI, 2007, p.50).

Belchior (2005, p.2) observa que o modelo DMAIC é fortemente utilizado por organizações desenvolvedoras de software. Pois, possuiu objetivo de fornecer subsídio para a melhoria contínua dos produtos desenvolvidos.

DMAIC é um modelo de melhoria continua para processos que necessitam de melhorias incrementais. Sendo este o mais adotado pelas empresas desenvolvedoras de software que usam o *Six Sigma* (STAMATIS (2004); TAYNTOR (2003) apud BELCHIOR, 2005, p.3). Ele corresponde a um acrônimo para as suas 5 (cinco) fases:

a) definir (***D****efine*);

b) medir (***M****easure*);

c) analisar (**A***nalyse*);

d) melhorar (**I***mprove*);

e) controlar (**C***ontrol*).

A figura 5 apresenta todas as fases do ciclo DMAIC.

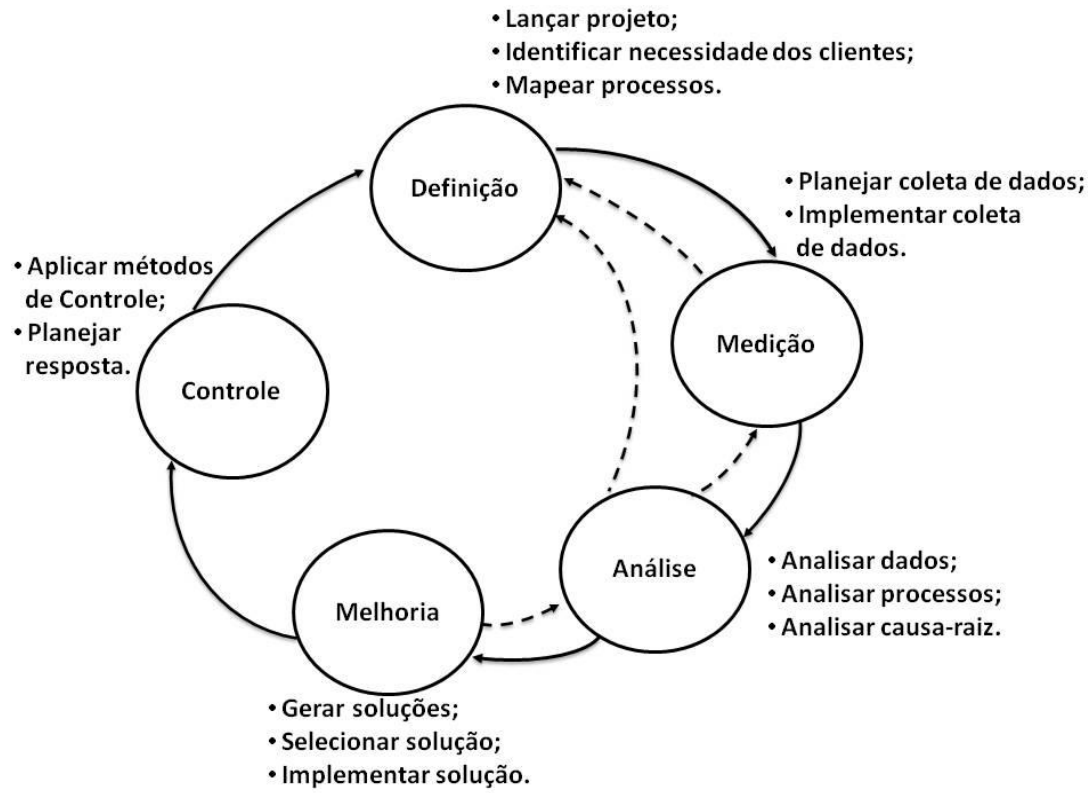


Figura 5 - Ciclo de melhoria DMAIC
Fonte: Eckes apud Aizawa (2007, p.21).

O quadro 4 mostra as principais atividades envolvidas para cada uma das fases.

Quadro 4 - Fases do ciclo DMAIC

Fase	Descrição
Definição	Esta fase possui como foco a identificação dos problemas e situações a serem melhoradas nos processos organizacionais de qualquer natureza, seja manufatureiro ou prestação de serviços. As melhorias identificadas através da análise dos processos organizacionais devem ter como foco principal o atendimento das necessidades dos clientes da organização. Os clientes da empresa são todos aqueles que utilizam o produto ou o serviço entregue, entre os clientes encontram-se os departamentos internos, funcionários e principalmente os clientes finais.
Medição	Na fase de medição, são discutidos os procedimentos para coleta de informações dos processos mapeados na fase de análise. É através da aplicação destes procedimentos, que as informações sobre o desempenho atual dos processos são identificadas. Estas informações são valiosas para o desenvolvimento do plano de coleta, permitindo com isto, preparar a estrutura de avaliação de desempenho dos processos.
Análise	O objetivo desta fase de análise é a consolidação do plano de coleta de informações da fase de medição e das oportunidades de melhoria identificadas na fase de definição. Esta análise em conjunto com as melhorias detectadas e das medições dos processos atuais, permite que a equipe de trabalho tire conclusões sobre as melhorias a priorizar, confirme sobre as reais necessidades de melhoria nos processos, identifique as origens dos problemas e, principalmente, quais são os reais benefícios das melhorias identificadas.
Melhoria	O objetivo desta fase é gerar idéias, desenhar programas de melhorias, realizar projetos pilotos de ajustes em processos e implementá-los. É através da análise dos resultados obtidos nas fases de definição, medição e análise que a fase de melhoria possui subsídios para propor mudanças e estar constantemente pensando em melhorias. A coleta de informações da satisfação dos clientes em conjunto com dados de desempenho de processos, auxilia a equipe de trabalho a propor mudanças, em alguns casos ajustes.
Controle	Esta fase do método tem como objetivo efetuar o controle nos processos existentes, aplicar as medições definidas com o intuito de monitorar o andamento dos processos e antecipar ações corretivas e de prevenção de desvio. Esta fase também é projetada para garantir que os ganhos obtidos nas fases anteriores sejam mantidos. Devem-se institucionalizar melhorias através de modificações em sistemas, estruturas e processos, tudo isto acompanhado por um plano de controle onde ficam registrados os responsáveis, o que está sendo mensurado, parâmetros de desempenho e medidas corretivas aplicadas.

Fonte: Adaptado de (STAMATIS apud AIZAWA, 2007, p.21-26).

O método de melhoria contínua DMAIC permite então o acompanhamento das métricas que podem ser definidas através do método GQM, que conforme Oliveira (2009, p. 86) é um método sistemático para definição e organização de indicadores, direcionados por metas pré-estabelecidas. A norma ISO/IEC 9126 fornece métricas destinadas a qualidade de software, a qual será tratada a seguir.

1.6 Série de normas ISO/IEC 9126

A norma ISO/IEC 9126 foi lançada em 1991 e submetida a uma revisão em 2001. Posteriormente, em 2003, resultou em uma nova versão. O grupo técnico responsável por elaborar a norma foi o subcomitê SC7 (Software e engenharia de sistemas), cujo os assuntos tratados são relacionados à Engenharia de Software e está subordinado ao (*Joint Technical Committee1*) JTC1, um comitê formado por membros das duas instituições ISO e IEC. Tem como objetivo subsidiar o processo de avaliação de produtos de software.

A norma ISO/IEC 9126 tem como título original *Software Engineering - Product Quality* que na versão brasileira, NBR ISO/IEC 9126, foi traduzido como Engenharia de Software - Qualidade de Produto (PINTO, 2008). A norma contém características e subcaracterísticas que definem um produto de qualidade. Sendo que está dividida desta forma:

- ISO/IEC 9126-1: Modelo de Qualidade;
- ISO/IEC 9126-2: Métricas Externas;
- ISO/IEC 9126-3: Métricas Internas;
- ISO/IEC 9126-4: Métricas de Qualidade em Uso.

No escopo da presente pesquisa, somente as métricas ISO/IEC 9126-2 (Métricas externas) e ISO/IEC 9126-3 (Métricas internas) serão estudadas, uma vez que abordam métricas para qualidade do software produzido.

1.6.1 Modelo de qualidade para qualidade interna e externa

Um modelo de qualidade para qualidade interna e externa, categoriza a qualidade de software em 6 (seis) características: 1) funcionalidade; 2) confiabilidade; 3) usabilidade; 4) eficiência; 5) manutenibilidade; e 6) portabilidade). Estas, por sua vez, são subdivididas em subcaracterísticas (figura 6). Para cada característica e sub-característica, a capacidade do software é determinada por um conjunto de atributos internos que podem ser medidos (ISO/IEC, 2003).

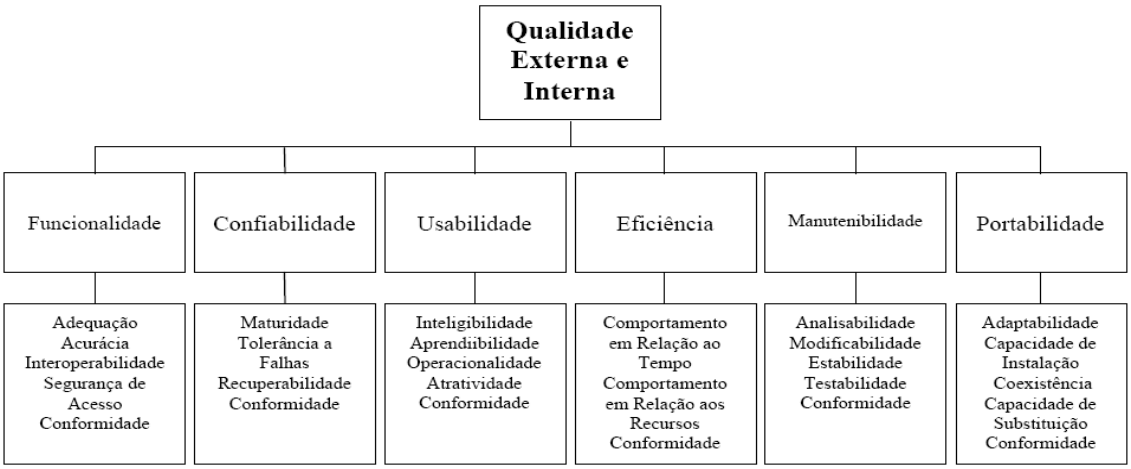


Figura 6 - Modelo de qualidade para qualidade externa e interna
Fonte: Adaptado de ISO/IEC (2003)

Cada característica e sub-característica está detalhada na sequência, de acordo com o quadro 5.

Quadro 5 - Características de métricas externas e internas

FUNCIONALIDADE	**Capacidade do software fornecer funções que correspondam às necessidades explícitas e implícitas do usuário quando o software é utilizado sob condições específicas. Essa característica se preocupa com o que o software faz para cumprir as necessidades do usuário.**
Adequação	Capacidade do software fornecer um conjunto apropriado de funções para tarefas especificadas e objetivos do usuário.
Acurácia	Capacidade do software fornecer o resultado com o grau de precisão desejado.
Interoperabilidade	Capacidade do software interagir com um ou mais sistemas.
Segurança de acesso	Capacidade de proteger dados e informações de pessoas ou sistemas não autorizados.
Conformidade	Capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a funcionalidade.
CONFIABILIDADE	**Capacidade do software manter seu nível de desempenho quando utilizado em condições estabelecidas.**
Maturidade	Capacidade de evitar defeitos no software.
Tolerância a falhas	Capacidade de manter um nível de desempenho estabelecido em caso de defeito no software.
Recuperabilidade	Capacidade de recuperar dados diretamente afetados no caso de falhas.
Conformidade	Capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a confiabilidade.
USABILIDADE	**Capacidade que o produto tem de ser entendido, aprendido, utilizado e ser atraente para o usuário.**
Inteligibilidade	Capacidade do produto de fazer o usuário entender se o software é adequado, e como ele pode ser usado para tarefas particulares.
Aprendibilidade	Capacidade que o produto deve ter de fazer o usuário entendê-lo.
Operacionalidade	Capacidade que o produto deve ter para que o usuário possa aprendê-lo e controlá-lo.
Atratividade	Capacidade do produto em ser atraente para o usuário.
Conformidade	Capacidade de aderir a padrões, convenções, leis e prescrições similares relativas a usabilidade.
EFICIÊNCIA	**Relacionamento entre o nível de desempenho do software e a quantidade de recursos utilizados, sob condições estabelecidas.**
Comportamento em relação ao tempo	Capacidade de fornecer tempos de resposta e processamento adequados, bem como taxas de transferência.
Comportamento em relação aos recursos	Capacidade de usar quantidade e tipos de recursos adequados.
Conformidade	Capacidade de aderir a padrões e convenções relativas a eficiência.
MANUTENIBILIDADE	**Capacidade do software ser modificado. Modificações podem incluir correções, aperfeiçoamentos ou adaptações do software a mudanças no ambiente, requisitos e especificações funcionais.**
Analisabilidade	Capacidade em diagnosticar deficiências e causas de defeitos.
Modificabilidade	Capacidade que o produto tem de receber modificações.
Estabilidade	Capacidade de evitar efeitos inesperados a partir de modificações.
Testabilidade	Capacidade de validar as modificações efetuadas no produto.
Conformidade	Capacidade de aderir a padrões e convenções relativas a manutenibilidade.
PORTABILIDADE	**Capacidade que o produto tem de ser transferido de um ambiente para outro.**
Adaptabilidade	Capacidade de ser adaptado em diferentes ambientes sem intervenção.
Capacidade de instalação	Capacidade de ser instalado em um ambiente específico.
Coexistência	Capacidade que o produto tem de coexistir com outro software independente em um ambiente comum, compartilhando recursos comuns.
Capacidade de substituição	Capacidade que o produto de software deve ter de ser usado no lugar de outro produto de software com o mesmo propósito no mesmo ambiente.
Conformidade	Capacidade de aderir a padrões e convenções relativas a portabilidade.

Fonte: Adaptado de Machado.

Para realizar a avaliação das características de qualidade externa são utilizadas as métricas externas; ou seja, medições baseadas nas necessidades do usuário: expostas na ISO/IEC 9126-2. De forma análoga, para a avaliação das características de qualidade interna são utilizadas as métricas internas: descritas na ISO/IEC 9126-3 (SODRÉ, 2006, p.26).

As métricas externas são usadas para avaliar o produto de software através de medições baseadas nas necessidades do usuário. Essas métricas usam medidas de um produto de software derivadas de medidas do comportamento do sistema do qual faz parte. As métricas internas avaliam a especificação ou o código fonte de um produto de software. Podem ser usadas também em partes intermediárias do produto em desenvolvimento para garantir qualidade final. O principal objetivo das métricas internas é assegurar a qualidade externa e a qualidade de uso.

Os anexos A e B exemplificam algumas métricas externas e internas da norma ISO/IEC 9126. Limita-se a algumas métricas devido ao fato de não ser de domínio público.

O estudo realizado a partir das referências bibliográficas sobre processo de desenvolvimento de software, métodos para definição de métricas e medidas para avaliar a qualidade de um produto, proporcionou conhecimento para construir um roteiro e servir como modelo de sugestão para definir métricas de software. Este modelo será apresentado na próxima seção.

2 ROTEIRO PROPOSTO

O modelo sugerido está baseado no método GQM, (conforme visto na seção 1.4.2). A escolha das métricas, num primeiro momento, passa por definir os objetivos. Após esta etapa concluída, elabora-se perguntas que atendam o objetivo a ser alcançado. Por último, embasado nas métricas estudadas propõe-se as que mais convêm de serem aplicadas para atender o objetivo completando assim o processo recomendado pelo método GQM fase definição[8] (figura 7).

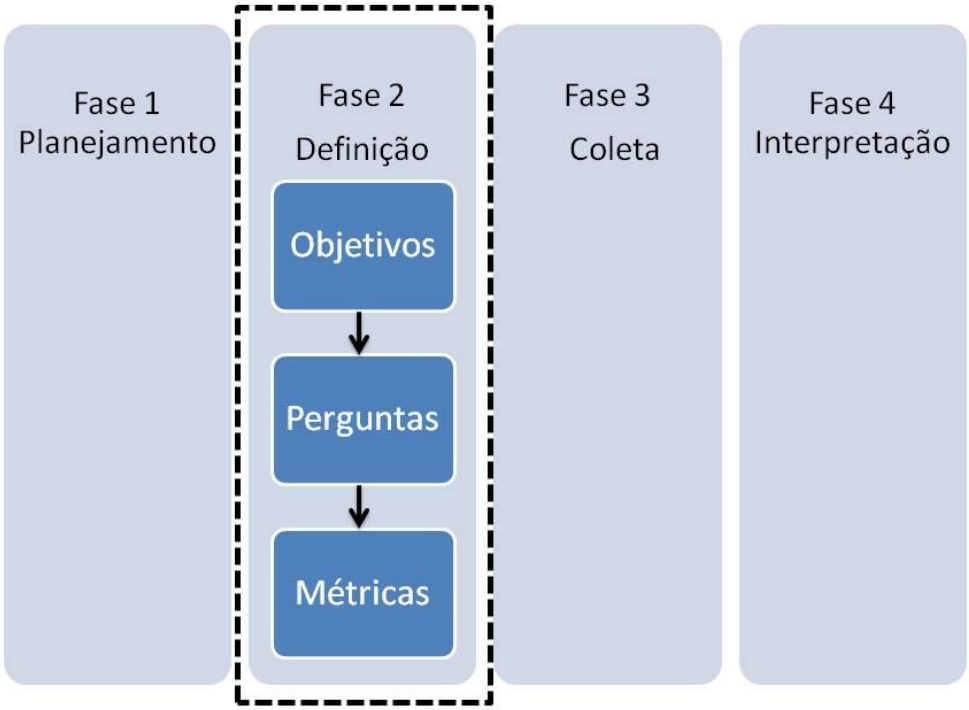


Figura 7 - Esquema GQM utilizado
Fonte: Adaptado de Solingen apud Covatti (2007, p.55).

Não é escopo desse trabalho as fases 1 (um), 3 (três) e 4 (quatro), visto que para definir as métricas propostas a fase de definição foi suficiente.

[8] As demais fases do método GQM já foram descritas anteriormente (seção 1.4.2).

A seção seguinte descreve a construção do roteiro elaborado.

2.1.1 Construção do roteiro

O objetivo final no processo de definir uma métrica é transformar dados em informações, mesmo que este dado seja classificado como qualitativo; ou seja, utilizar elementos textuais coletados e transformar em números, sendo possível assim efetuar comparações e, por consequência, a medição.

Ao iniciar o processo, primeiramente foram identificados os problemas a serem resolvidos, transformando estes problemas em objetivos a serem alcançados. Normalmente tem-se a necessidade de obter informações a respeito de algo; porém, existe a dificuldade em se chegar a uma métrica. Como exemplo de um objetivo pode se usar: ***"software oferecer segurança em relação aos dados entrados no sistema"***.

Após ter escolhido os objetivos a serem alcançados, elabora-se perguntas que possam encontrar uma ou mais métricas como resposta. Ao efetuar a pergunta para o objetivo anteriormente descrito, pode se utilizar: ***"qual o percentual de campos validados durante o processo de entrada de dados?"***.

Conforme apresentado anteriormente na seção 1.3, a métrica é geralmente composta por uma ou mais medidas. Para responder a pergunta elaborada, e conseguir atingir o objetivo proposto, escolhe-se 2 (duas) medidas: 1) quantidade de campos no software que permite a entrada de dados; e 2) quantidade de campos que possui validação não permitindo a entrada de dados incorretos. Ao possuir as duas medidas, utiliza-se a fórmula X = A / B, onde X será o resultado, A é a quantidade total de campos e B a quantidade de campos com validação. O resultado apurado (X) será a medida A dividida pela medida B, o valor de X, quanto mais próximo de 1 (um) melhor está alcançado o objetivo estipulado.

No modelo proposto nesse trabalho, para as métricas de qualidade foi utilizado a ISO/IEC 9126 que relaciona uma série de métricas de qualidade que podem ser utilizadas para alcançar os objetivos. Esta norma lista vários tipos de métricas que quando implementadas, acompanhadas, viabilizam obter qualidade do produto de software.

A busca pela qualidade deve ser um processo contínuo. Mesmo após definido os objetivos e as métricas, o processo deve ser acompanhado procurando identificar às oportunidades de melhorias, revendo os objetivos e, consequentemente, as métricas. O

método DMAIC sugere etapas para fazer o acompanhamento do processo, sendo este o método proposto.

Utilizou-se uma parte do método DMAIC na elaboração do modelo sugerido no presente estudo. Na primeira etapa (Definir), definiu-se os objetivos, as perguntas e as métricas apropriadas. Na etapa seguinte (Medir), é feito a coleta das informações referentes a cada métrica utilizada. A aplicação do questionário elaborado para validar o modelo sugerido foi entendido como sendo o cumprimento da etapa medir. Na terceira etapa (Análise), avaliou-se as respostas obtidas em relação ao modelo proposto, elaborado a partir do referencial teórico. As etapas 4 (Melhoria) e 5 (Controle) não fizeram parte do escopo do estudo em questão. As mesmas terão validade quando aplicados em uma organização que poderá a partir da etapa análise gerar soluções de melhoria e, então, aplicar os controles necessários.

A figura 8 ilustra o modelo proposto como sugestão para elaboração de métricas, utilizando os seguintes métodos: definição de métricas (GQM); e melhoria contínua (DMAIC).

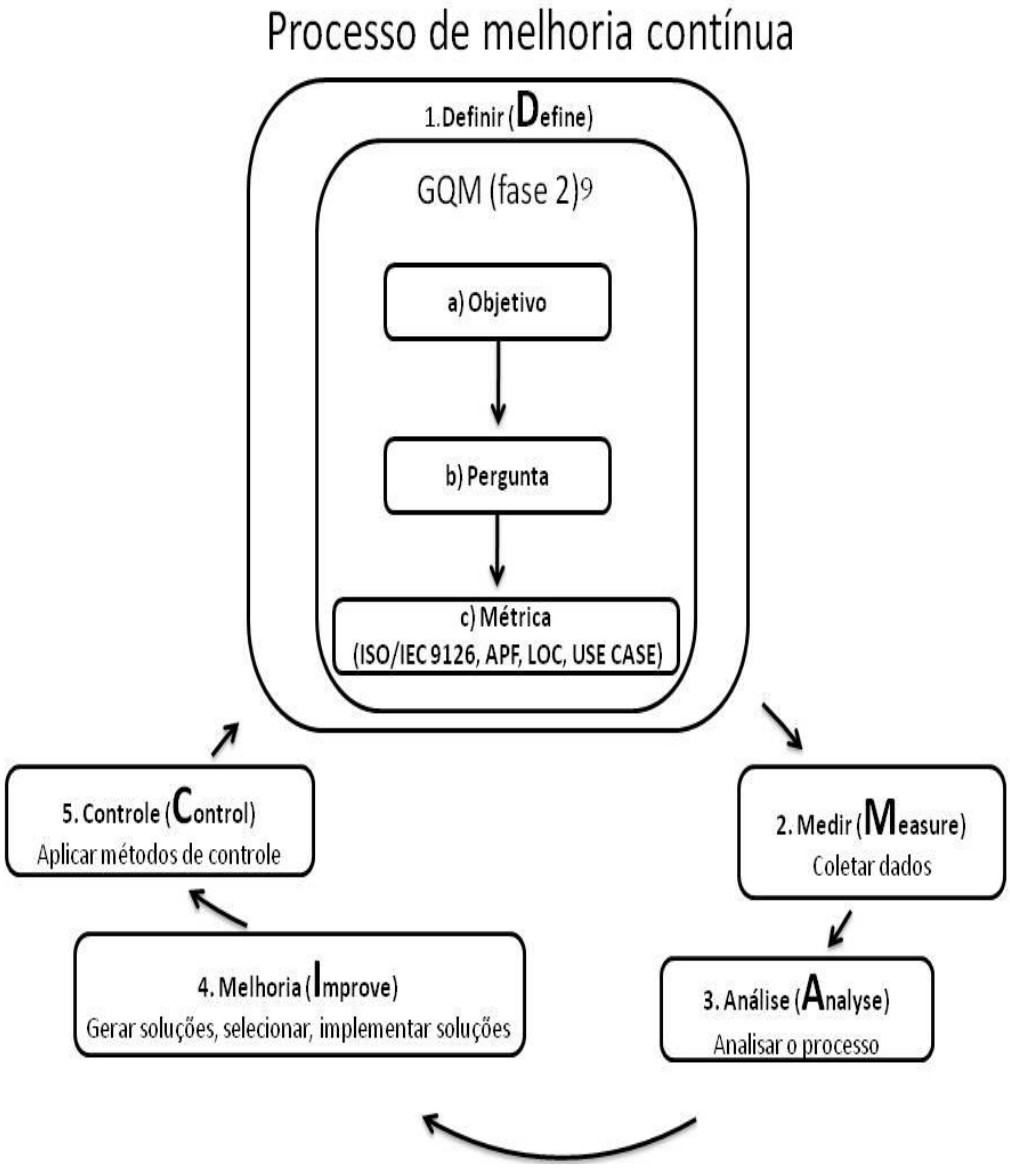


Figura 8 - Esboço do modelo proposto
Fonte: Adaptado de Aizawa; Covatti (2010).

Como forma de exemplificar o resultado proporcionado pelo modelo (roteiro), foi desenvolvido o quadro 6 que segue.

[9] Figura 7, ver página 35.

Quadro 6 - Métricas propostas

OBJETIVOS	PERGUNTAS	MÉTRICAS
1. Possibilitar medir o esforço no desenvolvimento novo ou manutenção de software.	1.1 Qual técnica mais adequada para medir o esforço de desenvolver software?	1.1.1 Análise por ponto de função (APF).
2. Entrega do software de acordo com os requisitos especificados.	2.1 Todos os requisitos foram implementados? 2.2 Os requisitos especificados foram implementados corretamente?	2.1.1 Cobertura de implementação funcional. 2.2.1 Adequação de implementação funcional.
3. Garantir que requisitos de precisão forneçam resultados corretos.	3.1 Os requisitos de exatidão especificados estão sendo atendidos? 3.2 Os requisitos de exatidão estão fornecendo informações corretas?	3.1.1 Precisão computacional. 3.2.1 Precisão.
4. Fornecer informações consolidadas entre os diversos sistemas.	4.1 As interfaces de dados implementadas estão de acordo com o especificado?	4.1.1 Coerência nas interfaces.
5. Acesso ao sistema somente a usuários autorizados.	5.1 O sistema permite que usuários indevidos acessem o sistema?	5.1.1 Controlabilidade de acesso.
6. Sistema executar as funcionalidades de maneira correta.	6.1 O sistema está apresentando falhas além do estimado?	6.1.1 Detecção de falhas.
7. Independência do usuário, na sua operação, para com a área de sistemas.	7.1 O usuário necessita de ajuda para operar o sistema?	7.1.1 Documentação de ajuda ao usuário.
8. Estabilidade no software.	8.1 As customizações efetuadas no software geram demandas de correções após as implementações?	8.1.1 Impacto da mudança.
9. Funções aptas a serem desfeitas.	9.1 O sistema permite que funções efetuadas possam ser desfeitas?	9.1.1 Desfazer uma operação.

Fonte: Autoria própria, (2010).

Visualmente, a figura 9 apresenta a relação entre as métricas sugeridas com as perguntas formuladas para atingir os objetivos. A maioria das métricas tem embasamento no modelo de qualidade da ISO/IEC 9126. Sendo exceção a 1.1.1, que é sugerida pelo (*International Function Point User Group*) IFPUG para medir software através de pontos de função.

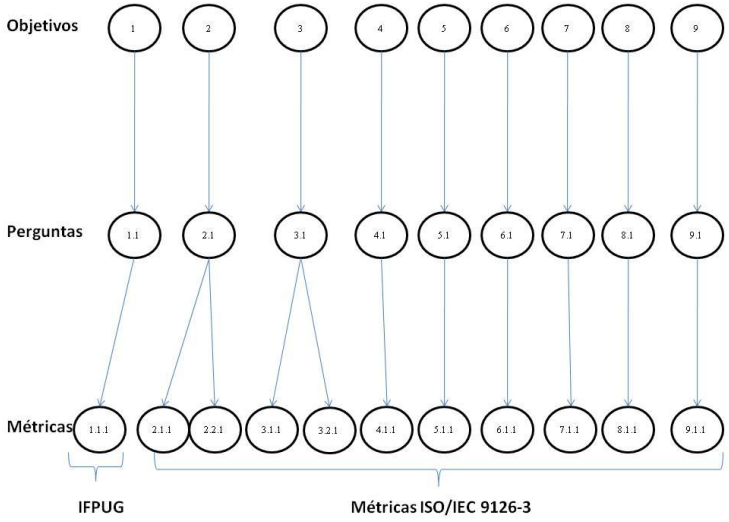


Figura 9 - Estrutura utilizada para definir métricas (GQM)
Fonte: Adaptado de Solingen apud Covatti (2007, p.55).

Ao usar o modelo sugerido (figura 8, página 37), definiu-se objetivos considerados pelo autor como sendo importante para um produto de qualidade, resultando as métricas para atender cada um dos objetivos (quadro 6). Toma-se como exemplo a segunda linha (2.1.1) desse quadro:

- passo 1, definiu-se o objetivo: ***entrega do software de acordo com os requisitos especificados;***
- passo 2, questionou-se como atender o objetivo através das perguntas:
    - ***2.1 – todos os requisitos foram implementados?; e***
    - ***2.2 – os requisitos especificados foram implementados corretamente?.***
- passo 3, pesquizou-se no referencial teórico (ISO/IEC 9126-3) as métricas que poderiam responder as perguntas, onde definiu-se as de números:
    - ***2.1.1 – cobertura de implementação funcional; e***
    - ***2.2.1 – adequação de implementação funcional.***

- passo 4, após identificar as métricas apropriadas, elaborou-se o quadro 7 com a definição de como apurar cada uma das métricas selecionadas.

Destaca-se que os objetivos listados podem ser diferentes de uma organização para outra. Cabe salientar que definir métricas visando atingir qualidade em software vai depender do que a organização vai definir por qualidade. A qualidade é relativa. O que é qualidade para uma pessoa pode ser falta de qualidade para outra (G. Weinberg).

O presente trabalho utilizou métricas pertencentes ao modelo de qualidade interna, uma vez que se tem como objetivo identificar métricas voltadas a melhorar o processo de desenvolvimento identificando rupturas na construção de software. A ISO/IEC 9126 – 2 (métricas externas) e 9126-4 (qualidade de uso) não fizeram parte – nesse estudo – da avaliação no momento da busca por métricas. A estrutura sugerida está descrita a seguir.

2.1.1.1 Estrutura das métricas

No exemplo sugerido, as métricas utilizadas são apresentadas no formato:

a) ***nome*** – nome da métrica;

b) ***propósito*** – é expresso como uma pergunta a ser respondida com a utilização da métrica;

c) ***método de aplicação*** – define a forma de medir a métrica;

d) ***medida e fórmula*** – exibe a fórmula e explica os dados utilizados nela;

e) ***interpretação*** – interpretação dos valores obtidos com o uso de intervalo de valores e os valores preferenciais;

f) ***tipo de escala métrica*** – tipo de escala utilizada pela métrica.

Os quadros 7 e 8, exemplificam como apurar o valor de cada métrica que foi definida para os objetivos identificados pelo autor do trabalho, expondo sua característica e sub-característica. Salienta-se que as métricas podem ser utilizadas de diversas fontes, como exemplo: para medir esforço de desenvolvimento foi usado análise por ponto de função; e quanto à qualidade utilizou-se métricas da ISO/IEC 9126 que trata da qualidade de software.

Quadro 7 - Métrica selecionada quanto a esforço

Métrica figura 9	Nome	Propósito	Método de aplicação	Medida e fórmula	Interpretação	Tipo de escala
1.1.1	Pontos de função.	Quantos pontos de função a aplicação possui?	Contar o número de EE, SE, CE, ALI, AIE (conforme seção 1.3.2).	X = A+B+C+D+E A = Quantidade de EE B = Quantidade de SE C = Quantidade de CE D = Quantidade de ALI E = Quantidade de AIE	X representa a quantidade de pontos de função de esforço para o desenvolvimento.	Absoluta

Quadro 8 - Métricas selecionadas quanto a qualidade

Característica – Funcionalidade						
Sub-característica - Adequação						
2.1.1	Cobertura de implementação funcional.	As funcionalidade especificadas estão implementadas?	Contar o número de funções faltantes detectadas e comparar com o numero de funções especificadas.	X = A/B A = número de funções implementadas. B = número de funções descritas na especificação.	0 <= X <= 1 Quanto mais próximo de 1, mais completo.	Absoluta
2.2.1	Adequação de implementação funcional.	As funcionalidades estão implementadas corretamente?	Contar o número de funções inadequada e comparar com o número de funções especificadas.	X = A/B A = número de funções com problemas. B = número de funções descritas na especificação.	0 <= X <= 1 Quanto mais próximo de 1, mais adequado.	Absoluta
Sub-característica - Exatidão						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
3.1.1	Precisão computacional	Os itens de precisão estão implementados em relação ao especificado?	Contar o numero de funções de precisão implementada e comparar com o número da especificação.	X = A/B A = número de funções com problemas. B = número de funções descritas na especificação.	0 <= X <= 1 Quanto mais próximo de 1, mais completo.	Absoluta
3.2.1	Precisão	Os requisitos de precisão estão fornecendo informações corretas?	Contar o número de requisitos que implementem precisão e comparar com o número de requisitos que requerem	X = A/B A = número de requisitos com problemas. B = número de funções descritas na especificação.	0 <= X <= 1 Quanto mais próximo de 1, mais adequado.	Absoluta

			precisão.			
Sub-característica - Interoperabilidade						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
4.1.1	Consistência nas interfaces	Os protocolos de interfaces estão implementados corretamente?	Contar o número de protocolos que foram implementados corretamente e comparar com o número de protocolos a serem implementados da requisição.	X = A/B A = número de protocolos implementados corretamente. B = número de protocolos solicitados na requisição	0 <= X <= 1 Quanto mais próximo de 1, mais consistente.	Absoluta
Sub-característica – Métricas de Segurança						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
5.1.1	Controlabilidade de acesso	O sistema tem controle de acesso?	Contar o número de requisitos de controle de acesso implementados e comparar com o número de requisitos solicitados na requisição.	X = A/B A = número de controles de acessos implementados corretamente. B = número de controle de acessos solicitados na requisição	0 <= X <= 1 Quanto mais próximo de 1, mais controlável.	Absoluta
Característica – Confiabilidade						
Sub-característica - Maturidade						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
6.1.1	Detecção de falhas	Como as falhas são detectadas na revisão do produto?	Contar o número de falhas detectadas e comparar com o número de falhas estimada a ser detectada nesta fase.	X = A/B A = número de falhas detectadas. B = número de falhas estimada para esta fase.	0 <= X Um valor alto para X implica um produto de boa qualidade.	Absoluta
Característica – Compreensibilidade						
Sub-característica – Métrica de aprendizibilidade						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
7.1.1	Documentação de ajuda ao usuário.	O sistema possui funcionalidade de ajuda ao usuário?	Contar o número de funções com documentação e comparar com o total de funções dos requisitos.	X = A/B A = número de funções descritas. B = número total de	0 <= X <= 1 Quanto mais próximo de1,mais completa.	Absoluta

				funções previstas.		
Característica – Manutenibilidade						
Sub-característica - Estabilidade						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
8.1.1	Impacto da mudança.	Qual é o impactos no software após mudanças?	Contar o número de efeitos adversos detectados após a modificação e compará-lo com o número de modificações realiza-das.	X = 1-A / B A = Número de efeitos adversos detectados após modificações. B = Número de modificações feitas.	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Absoluta
Característica – Operabilidade						
Sub-característica - Operacionalidade						
Métrica figura 9	**Nome**	**Propósito**	**Método de aplicação**	**Medida e fórmula**	**Interpretação**	**Tipo de escala**
9.1.1	Desfazer uma operação.	Qual a proporção de funções que podem ser desfeitas?	Contar o número de funções implementadas que o usuário pode desfazer após a conclusão e compará-lo com o número de funções total do sistema.	X = A / B A = Número de funções implementadas que póde ser desfeita pelo usuário. B = Número de funções do sistema.	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Absoluta

Fonte: ISO/IEC 9126-3.

3 METODOLOGIA

Conforme Roesch (2006, p.125), o capítulo da metodologia descreve como o projeto será realizado. Este capítulo tem o objetivo de expor o método utilizado com forma de pesquisa, as técnicas de coletas e análises de dados em que a presente pesquisa foi desenvolvida.

3.1 Método de pesquisa

Diferentes estratégias são abordadas para resolução de diversos problemas. Em qualquer tipo de estudo, pode-se valer de métodos quantitativos e qualitativos. Nos modelos quantitativos, procura-se trabalhar com base em amostras. Nas pesquisas qualitativas, busca-se fazer análise de profundidade tendo como referência a própria teoria.

A pesquisa a ser adotada será de natureza descritiva. Conforme Gil (2006, p.42), as pesquisas descritivas têm como objetivo primordial a descrição das características de determinada população ou fenômeno, ou, então, o estabelecimentos de relações entre variáveis. São inúmeros os estudos que podem ser classificados sob este título e uma de suas características mais significativas está na utilização de técnicas padronizadas de coleta de dados, tais como questionários e a observação sistemática.

Pozzebon; Freitas (1998, p.144) entendem que, entre os métodos qualitativos, pode-se destacar a importância potencial que o estudo de caso e a pesquisa-ação podem vir a desempenhar na área de sistemas de informação. Pesquisa-ação foi definida por Thiollent apud Gil (2007, p.46) como “um tipo de pesquisa social com base empírica que é concebida e realizada em estreita associação com uma ação ou com a resolução de um problema coletivo e no qual os pesquisadores e participantes representativos da situação ou do problema estão envolvidos de modo cooperativo ou participativo.” Pesquisa-ação consiste de um time de profissionais, que planejam, executam e avaliam os resultados obtidos a partir das ações que foram tomadas, e ainda monitoram as atividades. Fazem isso repetidamente até que os resultados se tornem satisfatórios (HOLANDA; RICCIO, 2010, p.4).

O presente estudo tem como foco propor métricas que possam ser aplicadas no processo de desenvolvimento de software, a sugestão será efetuada considerando as respostas obtidas através da pesquisa-ação. Benbasat; Goldstein; Mead apud Pozzebon; Freitas (1998, p.147) classificam pesquisa-ação quando o autor participa da implementação de um sistema e, simultaneamente, realiza certa intervenção técnica. Existe uma intenção original de

desenvolver uma pesquisa. O pesquisador possui 2 (dois) objetivos: 1) agir para resolver determinado problema; e 2) contribuir para um conjunto de conceitos em Sistemas de Informação.

3.2 Definição da área alvo da pesquisa

A organização selecionada é do ramo calçadista com 62 anos de existência, a empresa atua em diversos ramos de atividades, entre eles fabricação de calçados, empreendimentos, pecuária e varejo de calçados. Sendo que, para área alvo da pesquisa foi escolhida o ramo do varejo fundado em 1964. Presente no sul do Brasil e também no Nordeste Brasileiro a empresa possui em torno de 150 lojas de calçados distribuídas em 6 (seis) bandeiras, as quais identificam as suas lojas. Atualmente a empresa (varejo) conta com mais de 3.000 colaborados, sendo que na área administrativa são em torno de 250 colaboradores.

A escolha da organização para aplicação da pesquisa-ação foi definida em função dos seguintes itens:

- identificação de uma possibilidade de melhora no processo atualmente existente;
- facilidade de obtenção dos dados a serem coletados;
- conhecimento por parte do autor do processo atual.

3.2.1 Seleção dos sujeitos

Para Vergara (2007, p.53), os sujeitos da pesquisa são as pessoas que irão fornecer os dados necessários, aqueles que irão responder o questionário elaborado para efetuar a pesquisa-ação. Assim sendo, o presente estudo apresenta como sujeitos:

- o Gerente de (Tecnologia da Informação) TI;
- 3 (três) Analistas de Negócios; e
- 1 (um) Analista de Sistemas.

Os entrevistados foram selecionados considerando suas características e conhecimento, de modo que pudessem contribuir para o processo da pesquisa.

3.3 Coleta de dados

O questionário é um instrumento utilizado para coleta de dados formado por uma série ordenada de perguntas, que devem respondidas por escrito e sem a presença do

entrevistador (LAKATOS, 1999, p.100). Roesch (2006, p.140) entende que o questionário e a entrevista são as principais técnicas de coletas de dados, sendo o questionário o instrumento mais utilizado em pesquisa quantitativa.

A elaboração do questionário teve como base a análise de conteúdo do referencial bibliográfico pesquisado. Análise de conteúdo foi definida por Berelson como "uma técnica de pesquisa para a descrição objetiva, sistemática e quantitativa do conteúdo evidente da comunicação". Essa técnica permite analisar o conteúdo de livros, revistas, jornais, discursos, películas cinematográficas, propaganda de rádio e televisão, etc. (BERELSON apud GIL (2007, p.165); LAKATOS (1999, p.130-131)).

A elaboração do questionário viabiliza responder o objetivo geral e os específicos. Portanto, consiste em traduzir os objetivos em questões a serem respondidas. As questões constituem, pois, o elemento fundamental do questionário (GIL, 2007, p.129). O mesmo deve ser limitado em extensão e finalidade. Se for muito extenso causa desinteresse por parte da pessoa que irá respondê-lo, se curto demais, corre o risco de não oferecer suficientes informações. Lakatos (1999, p.101) recomenda que deva conter de 20 (vinte) a 30 (trinta) perguntas e que não deve demorar mais de 30 (trinta) minutos para ser respondido. Este número não é fixo, vai depender do tema da pesquisa e dos informantes. Corroborando com esta idéia, Gil (2007, p.134) salienta que alguns autores estabelecem como regra geral que o número de questões de um questionário não deve ultrapassar a 30 (trinta).

3.3.1 Etapas da pesquisa

Conforme Gil (2007, p.47-48), todo processo de pesquisa social envolve: planejamento; coleta de dados; análise e interpretação; e redação do relatório. Até o momento, não foi possível definir um modelo que apresente de forma absolutamente precisa e sistemática, os passos a serem observados no processo de pesquisa. Dentre as etapas sugeridas pelo autor, destaca-se as partes abaixo como integrantes de uma pesquisa.

3.3.1.1 Formulação do problema

De acordo com Gil (2007, p.45-54), na acepção científica, problema é qualquer questão não resolvida e que é objeto de discussão, em qualquer domínio do conhecimento. Diz ainda que o problema deve ser formulado como uma pergunta. Perante isso, pode-se dizer que o problema desta pesquisa-ação consiste em: ***há motivação em utilizar métricas no processo de desenvolvimento de software?***

3.3.1.2 Construção de hipóteses ou determinação dos objetivos

Hipótese é uma suposta resposta ao problema a ser investigado. É uma proposição que se forma e que será aceita ou rejeitada somente depois de devidamente testada (GIL, 2007, p.56). Lakatos (1999, p.30) ainda define hipótese como sendo uma suposição que antecede a constatação dos fatos e tem como característica uma formulação provisória. As suposições que espera-se comprovar através da pesquisa-ação elaborada neste trabalho e que podem melhorar o processo de desenvolvimento do software são:

- para o ambiente de desenvolvimento estudado a técnica da análise ponto de função é a recomendada;
- é necessário ter método de medição de esforço na construção de um software;
- programas de certificações garantem um produto de qualidade;
- métricas devem somente ser determinadas quando se tem um objetivo claramente definido e entendido.

3.3.1.3 Plano de coleta de dados

Vergara (2007, p.54) orienta que ao efetuar uma coleta de dados, o leitor deve ser informado de como o entrevistador pretende obter as informações que precisa para conseguir responder o problema.

O questionário (ANEXO G) foi elaborado a partir de um estudo da bibliografia existente sobre métricas, qualidade de software, processo de desenvolvimento e melhoria contínua, composto com perguntas abertas e fechadas. O questionário caracteriza-se por uma série de questões apresentadas ao respondente, por escrito. Pode ser aberto, pouco ou não estruturado, ou fechado, estruturado. No questionário aberto, as respostas livres são dadas pelos respondentes; no fechado, o respondente faz escolhas, ou pondera, diante das alternativas apresentadas (VERGARA, 2007, p.54-55). Para Roesch (2006, p.143), as perguntas podem ser fechadas ou abertas, ou alguma combinação entre elas. Esta última opção foi adotada no questionário deste trabalho.

Todas as questões que compõe a pesquisa foram aplicadas a todos os sujeitos relacionados anteriormente na seção 2.2.1. Num primeiro momento, foi realizado um pré-teste com 8 (oito) alunos de uma disciplina do curso de graduação Sistemas de Informação

(Universidade Feevale[10]). O pré-teste foi realizado com o objetivo de assegurar a clareza das questões e objetividade das respostas. O mesmo, foi composto por 19 (dezenove) questões. Para Lakatos (1999, p.102), o pré-teste serve para verificar se o questionário apresenta 3 (três) importantes elementos: 1) fidedignidade – qualquer pessoa que o aplique obterá sempre os mesmos resultados; 2) validade – os dados recolhidos são necessários à pesquisa; e 3) operatividade – vocabulário acessível e significado claro. A partir desta ação identificou-se as seguintes necessidades:

a) adequar questões que não tinham clareza;

b) refazer as que forneceram respostas dúbias;

c) inclusão de questões que atendessem aos objetivos do estudo.

O questionário final é composto por 22 (vinte e duas) questões (ANEXO G). O mesmo foi elaborado da seguinte maneira (figura 10):

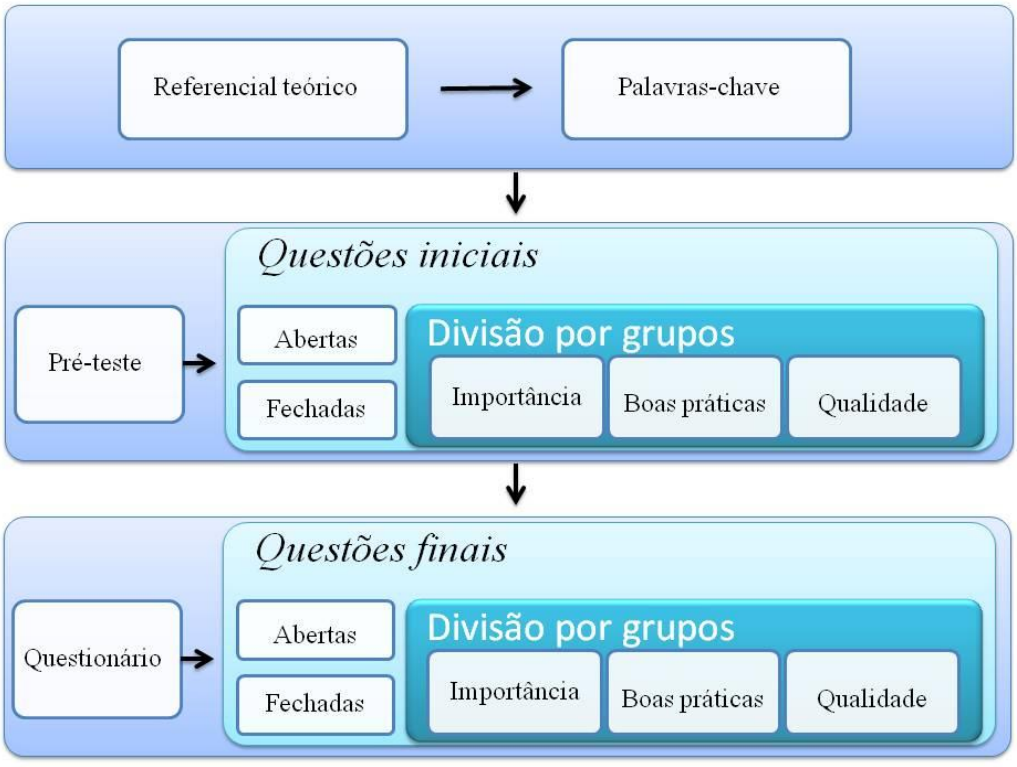


Figura 10 - Esquema para coleta de dados
Fonte: Autoria própria, 2010.

d) a partir do referencial teórico, pesquisou-se palavras-chave de maior relevância, que possibilitou alcançar os objetivos específicos;

[10] Universidade brasileira, localizada no município de Novo Hamburgo, na região metropolitana de Porto Alegre, Estado do Rio Grande do Sul.

e) em seguida, com base nas palavras-chave definidas, preparou-se as questões iniciais para compor o pré-teste;

f) as questões foram formuladas no modo de perguntas abertas e fechadas;

g) na sequência, o pré-teste foi reavaliado posteriormente, foi definido o questionário. Composto – então – pelas questões finais com perguntas abertas de fechadas.

No Pré-teste e questionário final, as perguntas foram divididas em 3 (três) grupos[11]: 1) importância; 2) boas práticas; e 3) qualidade.

O grupo ***importância*** consiste de questões que abordam o valor que a medição tem para o processo de desenvolvimento de software. O segundo grupo, contempla questões relacionadas a ***boas práticas*** que o mercado de software utiliza na construção de sistemas. Por último, o terceiro grupo é constituído de perguntas referentes à ***qualidade***. As mesmas, foram elaboradas, em função do interesse em se conhecer a relevância que se dá a um produto com qualidade.

3.3.1.4 Plano de análise de dados

A partir dos dados obtidos através do questionário, utilizou-se a técnica de análise de conteúdo para responder ao seguinte objetivo específico: ***propor um roteiro para definir métricas***. Os dados quando analisados sob forma de um texto, ou de um conjunto de textos, ao invés de uma tabela com valores, a análise correspondente assume o nome de análise de conteúdo (FREITAS; JANISSEK, 2000, p.37). Para realizar a análise de conteúdo, os autores sugerem seguir as seguintes etapas: 1) definição do universo; 2) categorização do universo estudado; 3) escolha das unidades de análise; e 4) quantificação (figura 11).

[11] Ressalta-se que no questionário as perguntas não estão agrupadas (importância, boas práticas, qualidade), visto que o mesmo não interfere no resultado final.

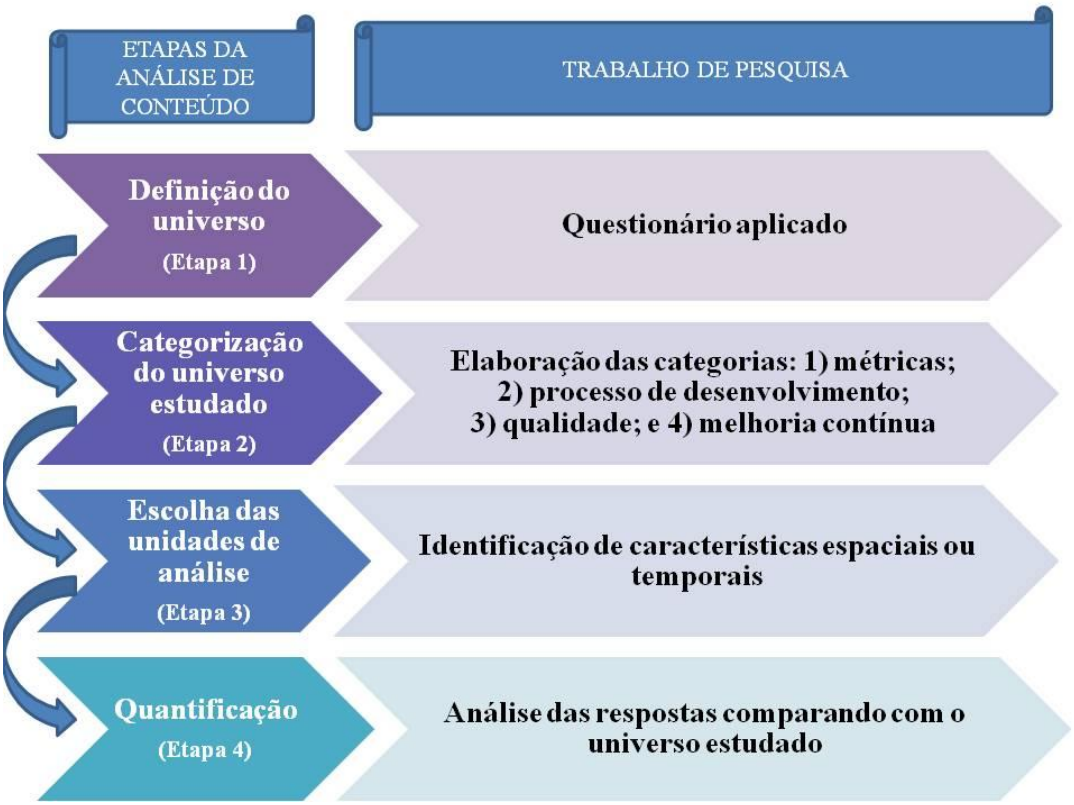


Figura 11 - Etapas da análise de conteúdo
Fonte: Adaptado de Freitas; Janissek (2000, p.45).

De acordo com a figura 11, a etapa 1 (***definição do universo***) delimita e define o universo a ser estudado. Apresenta o que está e o que não está na análise de conteúdo. O universo selecionado nesse estudo foi delimitado pelas respostas obtidas no questionário aplicado na pesquisa.

Após a conclusão da etapa 1 (um), elaborou-se a ***categorização do universo de estudo*** (etapa 2). Neste trabalho, a escolha das categorias se deu a partir de palavras-chave retiradas do questionário aplicado. No ponto de vista de Gil (2007, p.168-169), a análise tem como objetivo organizar e sumarizar os dados de tal forma que possibilitem o fornecimento de respostas ao problema proposto para investigação. As respostas fornecidas pelos elementos da pesquisa tendem a ser mais variados. Para que as respostas possam ser adequadamente analisadas, torna-se necessário organizá-las em grupos identificados como categorias. Aquelas que apresentaram as mesmas palavras-chaves foram agrupadas na mesma categoria. Assim sendo, as palavras-chave foram divididas em 4 (quatro) categorias (figura 12):

a) métricas;

b) processo de desenvolvimento;

c) qualidade;

d) melhoria contínua.

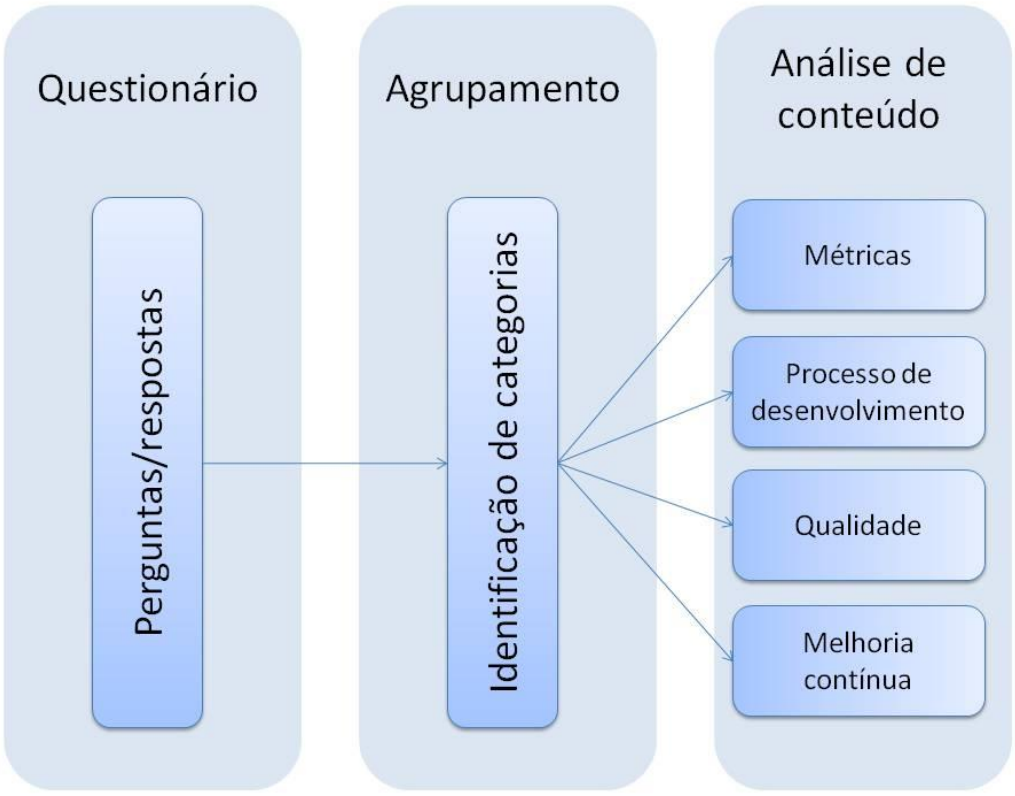


Figura 12 - Análise das respostas obtidas no questionário
Fonte: Autoria própria, 2010.

Na sequência da etapa 2 (categorização do universo estudado), foi efetuada a ***escolha das unidades de análise*** (etapa 3). Todas as categorias foram investigadas em profundidade, pois apresentam características espaciais ou temporais que na visão de Freitas; Janissek (2000, p.47) implicam em relacionar especificidades das respostas evidenciando o conjunto total das idéias apresentadas.

Na última etapa (***quantificação***), realizou-se a análise e a interpretação dos dados combinado com o universo estudado. As mesmas são apresentadas no capítulo que segue (capítulo 3).

4 ANÁLISE DE DADOS

Em uma pesquisa de caráter qualitativo, após o encerramento da coleta dos dados, o pesquisador se depara com uma grande quantidade de depoimentos, respostas em formato de texto, as quais terão que organizar para depois interpretar. Procura-se utilizar técnicas que seguem os padrões quantitativos, ou seja, tem o propósito de contar a frequência de um fenômeno. Costuma-se denominar o conjunto destas técnicas de análise de conteúdo (ROESCH, 2006). Neste capítulo, serão apontados os dados extraídos da pesquisa, descrevendo e analisando-os de forma a responder os objetivos específicos. A figura 13, ilustra a forma de como se tratou da apuração dos dados.

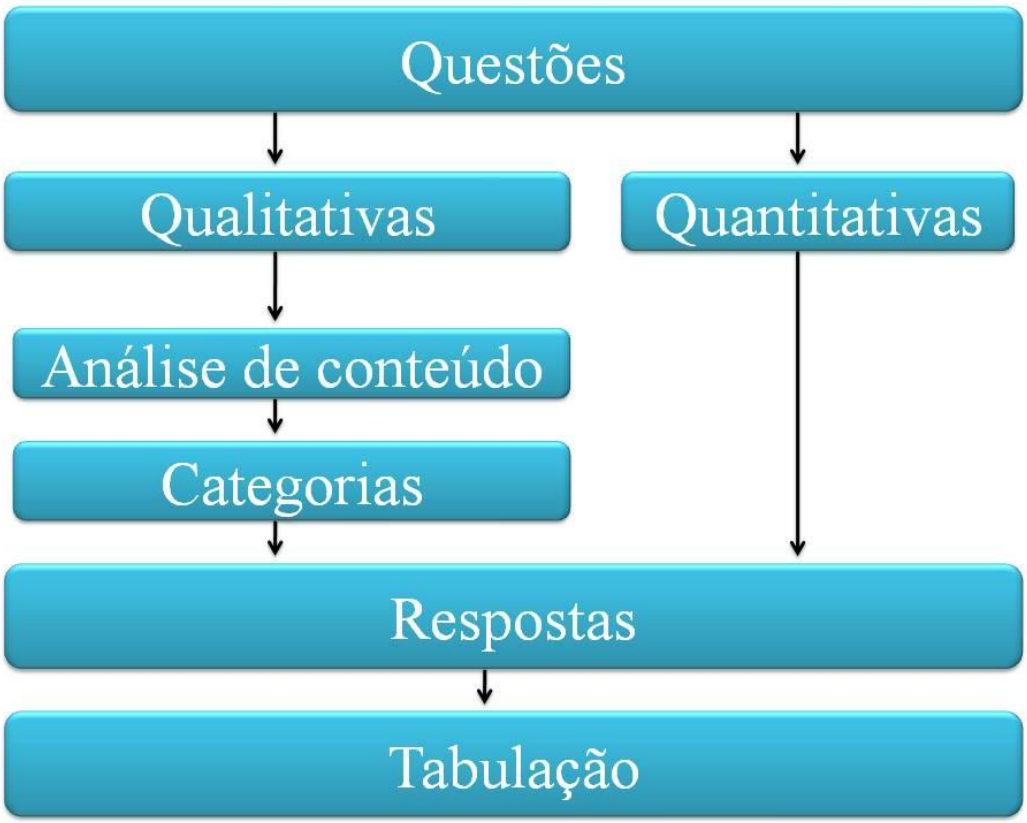


Figura 13 - Esboço da apuração dos dados
Fonte: Autoria própria, 2010.

Conforme descrito anteriormente, as questões do questionário são do tipo abertas (qualitativas) e fechadas (quantitativas). Para as qualitativas, aplicou-se a técnica análise de conteúdo; quanto às quantitativas, analisou-se as respostas através da tabulação dos dados.

4.1 Análise de conteúdo

A partir das respostas do questionário aplicado, identificou-se 4 (quatro) categorias (figura 12). Cada uma delas, foi dividida em 3 (três) outras categorias. Quais sejam: a) ***categoria inicial*** (composta pela síntese da questão); b) ***categoria intermediária*** (oriunda da resposta obtida); e c) ***categoria final*** (análise sucinta da resposta (ANEXOS C, D, E, F)). Para todas as questões pertencentes a cada categoria, elaborou-se uma análise das respostas fornecidas.

O quadro 9 exibe o resultado da análise de conteúdo efetuada na seguinte estrutura:

a) ***pergunta*** – Questões que compõe o questionário utilizado na pesquisa de validação;

b) ***análise*** – Síntese elaborada pelo autor a partir das respostas obtidas através do questionário;

c) ***RT*** – Referencial teórico que permite validar ou não, a análise efetuada em relação as respostas obtidas.

d) ***parecer*** – Em relação à empresa em estudo.

Quadro 9 - Análise qualitativa

Nº pergunta (Anexo G)	Estrutura de análise
1	**Pergunta:** Não se pode controlar o que não se mede (DEMARCO, 1982). Você considera esta afirmação válida quando o assunto é desenvolvimento de software? Justifique. **Análise:** Utilizar métricas em um ambiente de desenvolvimento de software permite efetuar controles no processo, sendo possível gerar histórico de informações afim de serem analisadas em um processo de melhoria contínua. **RT:** Quanto a esta questão, cita-se o que Pressman (1995, p.59) afirma: medir é fundamental em qualquer disciplina de engenharia, e a engenharia de software não é exceção. **Parecer:** ***Quanto a este conceito, chega-se a conclusão que a empresa (objeto do presente estudo) está consciente que medir é importante para desenvolver software.***
2	**Pergunta:** Você considera importante utilizar técnicas para medir o esforço no desenvolvimento de software? Por quê? **Análise:** Medir esforço no desenvolvimento de software proporciona fazer estimativas de entregas do produto, de modo que seja possível desta forma estabelecer um prazo de entrega considerando uma metodologia e não baseado apenas no "*feeling"* e/ou na experiência do analista de sistemas. **RT:** Peters (2001, p.430) entende que a métrica do software permite medir e prever os processos de software, os recursos necessários de um projeto e os artefatos relevantes ao esforço de desenvolvimento. **Parecer:** ***O processo de desenvolvimento da empresa não possui métricas formais. Sugere-se que seja implementada esta prática a fim que seja possível obter melhores resultados na qualidade do produto.***
3	**Pergunta:** Entre os tipos de medidas, análise pontos de função e linhas de códigos, qual tipo de medida você considera mais adequado a ser utilizado atualmente? **Análise:** Na pesquisa, a análise por pontos de função se mostrou mais conveniente. Foi avaliada como uma técnica que gera bons resultados, permite clareza entre o analista de sistemas e o usuário, pois efetua medição a partir das funcionalidades a serem desenvolvidas que são de conhecimento do usuário. **RT:** A análise de pontos de função é uma técnica para medir o tamanho funcional de um software do ponto de vista do usuário. Portanto, deve medir a funcionalidade que o usuário solicita e recebe, deve ser simples o suficiente para minimizar o trabalho adicional envolvido no processo de medição VAZQUEZ, 2008, P.52). **Parecer:** ***A partir do referencial teórico estudado e a análise das respostas obtidas permite concluir que a utilização de pontos de função no processo de desenvolvimento deve ser implementada. Em função de ser uma metodologia nova para a empresa, propõe-se que seja providenciado treinamentos aos envolvidos no processo.***
10	**Pergunta:** Segundo Koscianski (2007), ao definir uma determinada métrica, deve ser identificado primeiramente qual objetivo será alcançado. Desta mesma forma o processo de medição do MPSBR (nível F) sugere utilizar métodos que definem métricas orientadas a objetivos (SOFTEX, 2009). Você considera esta sugestão válida? Por quê?

	Análise: Considerou-se válido a escolha de métricas orientadas a objetivos pré-definidos. Desta forma, entende-se que existe certa facilidade em definir as métricas que auxiliarão o controle do processo. **RT:** A definição de métricas orientadas a metas ajuda a organização a focalizar as métricas adequadas para o negócio (PRESSMAN, 2010, p.516). **Parecer:** ***Neste quesito, como no caso anterior, recomenda-se um treinamento para orientar na definição das métricas a serem utilizadas. Como já citado (1.4.2), identificar as métricas a partir de um objetivo definido (etapa 2 do método GQM: figura 8, página 38).***
14	**Pergunta:** Na opinião de Pressman (2002), percebe-se que a cultura de medição de software não é comum em pequenas empresas de desenvolvimento de software. Em sua opinião, cite 2 (duas) razões porquê isto acontece. **Análise:** Dentre as razões apresentadas como resposta a esta pergunta, destaca-se: • A falta de conhecimento de como executar técnicas de medição; • Falta de tempo: entende-se que para efetuar medição consome-se tempo desnecessário; • Identificou-se ainda que a falta de recursos financeiros é também um dos motivos alegados. **RT:** Nesta análise, identificou-se uma relação entre a bibliografia estudada com a prática. **Parecer:** ***Quanto a esta questão, é essencial: identificar as pessoas que necessitam de treinamento em relação a utilização de métricas; avaliar entre as opções de métricas de esforço qual é a mais adequada; considerar a disponibilidade de tempo e de recursos financeiros.***
7	**Pergunta:** Empresas buscam certificações, pode-se considerar que ocorre ganhos nos processos de desenvolvimento utilizar estas certificações (CMMI, MPSBR)? **Análise:** Entre a maioria entrevistada, nota-se que há um consenso que o processo de certificações traz vantagens ao processo de desenvolvimento, melhorando a qualidade do produto. **RT:** Para Couto (2007, p.3), na globalização do mercado de software, terá mais chances de sobreviver quem for organizado e eficiente no seu processo de produção. Além dos benefícios naturais, como produtividade e qualidade, comercialmente acredita-se que, em curto prazo, a certificação dos processos fabris será um pré-requisito básico para as contratações de produto de software. **Parecer:** ***A empresa em questão, utiliza software desenvolvido por equipe própria e adquirido de terceiros. Quanto ao desenvolvimento próprio, o software é apenas para uso interno não caracterizando a sua comercialização consequentemente a implementação de métricas internas é de maior importância que um processo de certificação.***
12	**Pergunta:** Medições fornecem um feedback (retorno) para melhoria contínua do processo (PETERS, 2001). Você considera importante utilizar uma ferramenta que traz melhoria para processo de desenvolvimento? Por quê? **Análise:** No entendimento dos entrevistados, um processo de melhoria contínua facilita o controle e permite a partir das experiências implementar procedimentos que agregam qualidade ao produto. **RT:** Peters (2001, p.35) considera que no contexto dos sistemas de software em evolução, o *feedback* dos sistemas em operação, assim como as informações contidas em documentos de software existentes contribuem para o desenvolvimento de uma nova versão de software. **Parecer:** ***Sugere-se implementar um processo de melhoria contínua de forma padronizada (como por exemplo, o DMAIC).***

11	**Pergunta:** Em sua opinião, o que significa qualidade de software? **Análise:** O software deve atender os requisitos solicitados. Ser entregue ao cliente dentro do prazo acordado e ter processos bem definidos. Permitindo – assim – rastreabilidade através de documentação. **RT:** Segundo Pressman (2010, p.578), no desenvolvimento de software, a qualidade de projeto abrange os requisitos, as especificações e o projeto do sistema. A qualidade de conformidade é um assunto concernente, principalmente, à implementação. Se a implementação segue o projeto e o sistema resultante satisfaz os requisitos e metas desempenho, a qualidade de conformidade é alta. **Parecer:** ***Através da implementação de métricas (como, por exemplo, métricas do quadro 7 apresentado na página 41) e processos sugeridos (GQM e DMAIC) pressupõe-se que a empresa tenha um ganho na qualidade, permitindo assim a entrega do produto que atenda aos requisitos especificados pelo cliente.***
17	**Pergunta:** A norma ISO/IEC 9126 trata de um modelo de qualidade para um produto de software, que permite orientar diferentes perspectivas de avaliação. Por exemplo: desenvolvedores e clientes tem visões e preocupações diferentes relacionadas a qualidade do mesmo produto (KOSCIANSKI, 2007, p.206). Cite 2 (dois) motivos que possam justificar a afirmação acima. **Análise:** O cliente está preocupado que o software possua as funcionalidades que necessita para desempenhar o seu trabalho com facilidade de utilização. O desenvolvedor se dedica a desenvolver com um olhar voltado mais para as questões técnicas. **RT:** Conforme estudado, a qualidade depende do entendimento das partes envolvidas, para isso precisa ficar claro na especificação dos requisitos qual é a expectativa de cada parte envolvida em relação ao produto a ser desenvolvido. **Parecer:** ***No que diz respeito a documentação de requisitos, não foi possível comprovar através das respostas (questionário) a existência ou não desta prática. Recomenda-se então uma avaliação mais apurada deste tópico com o objetivo de comprovar ou não o uso deste procedimento.***

Fonte: Autoria própria, (2010).

4.2 Tabulação quantitativa

Dentre as questões quantitativas do questionário elaborado, as que tiveram uma maior relevância em termos do consenso das respostas foram as perguntas que seguem. O quadro 10 a seguir exibe o resultado da tabulação efetuada na seguinte estrutura:

e) ***Pergunta*** – Questões que compõe o questionário utilizado na pesquisa de validação;

f) ***Análise*** – Resultado considerando as respostas de maiores expressão.

g) ***Parecer*** – Em relação à empresa em estudo.

Quadro 10 - Análise quantitativa

Nº pergunta (Anexo G)	**Estrutura de análise**
4	**Pergunta:** Marque abaixo o(s) item(s) que você considera importante quando se fala em utilizar técnicas de medição de software em um processo de desenvolvimento. **Análise:** Os 5 (cinco) entrevistados entenderam que esta prática agrega valor ao produto. **Parecer:** ***Através das questões 4 e 6 fica nítida a necessidade de implementação de métricas no processo de desenvolvimento atualmente utilizado pela empresa em estudo.***
6	**Pergunta:** Considerando as definições apresentadas a respeito de medição de software, qual das alternativas a seguir você considera agregar mais valor. **Análise:** A opção escolhida por todos os entrevistados foi que a medição de software agrega valor ao processo de desenvolvimento como um todo. **Parecer:** ***Esta questão, reforça a sugestão já feita anteriormente, qual seja: métricas de software.***
15	**Pergunta:** Em sua opinião, além da construção de um sistema, quais atividades a baixo NÃO deveriam deixar de fazer parte do processo de desenvolvimento de software. **Análise:** A estimativa de esforço também se mostrou importante nesta questão. A mesma obteve votação de todos (5) respondentes. **Parecer:** ***Nota-se que conseguir estimar esforço na construção do software também é necessário. Assim, orienta-se que seja utilizado pontos de função para dimensionar o tamanho do esforço e com isso conseguir estimar prazos de entrega do produto.***
16	**Pergunta:** Para se obter uma medida de software existem diversas métricas. Marque na lista a seguir qual(is) você já ouviu falar ou teve algum contato. **Análise:** As opções selecionadas nesta questão foram APF (Análise de pontos de função) e UCP (*Use Case Points*). **Parecer:** ***O fato dos entrevistados conhecerem (ou ouviram falar) a respeito de APF, facilita a implementação desta metodologia.***
19	**Pergunta:** Atualmente você considera utilizar, usufruir de métricas que forneçam medidas de esforço e qualidade nos softwares desenvolvidos. **Análise:** A utilização de métricas no processo de desenvolvimento não é praticada de maneira satisfatória. **Parecer:** ***Com base na análise feita, chega-se a conclusão que é fundamental a implantação de uma política de métricas no processo que atualmente é utilizado.***

21	**Pergunta:** Em sua opinião, marque a(s) questão(s) que considera importante medir quando relacionado à qualidade de software. **Análise:** Dentre as opções listadas a opção utilizar métricas para obter controle se o sistema gera resultados corretos ou conforme acordado foi a mais utilizada por todos (5) respondentes. **Parecer:** ***Através da pesquisa é possível identificar o entendimento quanto as vantagens e necessidades da utilização de métricas em produto de software. Sendo assim, orienta-se iniciar um processo para identificar e implementar as métricas que sejam de maior relevância para a organização.***

Fonte: Autoria própria, (2010).

As demais questões (identificadas através dos números 5, 8, 9, 13 e 18) obtiveram respostas muito individualizadas, não mostrando consenso entre os respondentes o que não permitiu uma interpretação comum.

CONCLUSÃO

A área da TI nas companhias está sendo fundamental para que esta possa alcançar suas metas, independente de ser área fim ou meio. Nota-se que o software tem uma relevante contribuição neste processo. O software passou a ser um bem indispensável para empresas que procuram a satisfação do cliente. Essa necessidade faz com que as organizações invistam mais na qualidade de seus produtos de software, visando conquistar novos clientes e manter os já existentes.

A engenharia de software contribui para este processo ao se preocupar com a qualidade do desenvolvimento de software, identificando e sugerindo metodologias a serem aplicadas durante o processo do desenvolvimento.

Este trabalho de conclusão iniciou pelo estudo da bibliografia a respeito de métricas de software. Verificou-se no decorrer do estudo que a medição na área de software é importante para que o processo de desenvolvimento possa ser acompanhado. As fontes pesquisadas foram, além de livros, materiais disponíveis na internet.

A partir da bibliografia estudada, foi possível propor um roteiro para que seja viável definir métricas a fim de permitir dimensionar o esforço no desenvolvimento de software, quando relacionado a identificar estimativas de prazo. Ao mesmo tempo, também viabilizar a melhora do produto produzido através da aplicação de métricas que avaliam a sua qualidade. A principal contribuição do trabalho é o modelo (roteiro) que oportuniza definir métricas de acordo com os objetivos individuais de cada organização (uso do método GQM). O modelo contempla também utilizar etapas que tratam da melhoria contínua de um processo (modelo DMAIC).

A aplicação do questionário – pesquisa-ação – resultou em respostas que contribuíram para evidenciar as hipóteses propostas:

- Para o ambiente de desenvolvimento estudado a técnica da análise ponto de função é a recomendada;
    - Ficou evidenciado que a análise por pontos de função, é uma medição desejada pelos entrevistados.
- É necessário ter método de medição de esforço na construção de um software;

- Em meio a todas as respostas, foi intensamente almejado possuir métricas que possam apurar o esforço no desenvolvimento de software.

- Programas de certificações garantem um produto de qualidade;
    - A utilização de programas de certificação (com MPSBR, CMMI), neste estudo de caso, não obteve tal importância em função da utilidade do software para empresa em questão.

- Métricas devem somente ser determinadas quando se tem um objetivo claramente definido e entendido.
    - Foi possível identificar o desejo de obter métricas a partir da definição de um objetivo.

A limitação do trabalho desenvolvido está por conta que a sua verificação quanto à aderência do modelo foi focada em uma única empresa, onde foi efetuado uma pesquisa-ação com o interesse de implementar o modelo sugerido no trabalho em relação à empresa objeto de estudo.

Sugere-se para trabalhos futuros:

- Que o modelo em questão possa ser aplicado em diversas empresas. Tanto em empresas que desenvolvem software para uso interno, como para àquelas que comercializam aplicativos (“software house”);

- Aplicar o modelo proposto na empresa estudada, a fim de definir as métricas necessárias que possam contribuir para a melhora na qualidade do produto desenvolvido.

REFERÊNCIAS BIBLIOGRÁFICAS

AIZAWA, Maurício. **Um comparativo entre as abordagens seis sigma e GQ(I)M focado em melhoria dos projetos de software.** Disponível em: <http://www.sscontrols.com.br/maizawa/DissertacaoFinal_MauricioAizawa.pdf>. Acesso em: 25 maio 2010.

ASSOCIAÇÃO PARA PROMOÇÃO DA EXCELENCIA DO SOFTWARE BRASILEIRO – SOFTEX. **MPS.BR – Guia de implementação 2009**, maio 2009. Disponível em www.softex.br.

BELCHIOR, Arnaldo D. **Métricas de software:** um mapeamento entre Six Sigma e CMMI. Disponível em: <http://www.simpros.com.br/simpros2005/upload/A10_3_ artigo14725.pdf>. Acesso em: 20 junho 2010.

COUTO, Ana B. **Cmmi:** Integração dos modelos de capacitação e maturidade de sistemas. Rio de Janeiro: E. Ciência Moderna, 2007.

COVATTI, Andressa. **MIBCIS – Método de integração entre o BSC, CMMI e Six Sigma utilizando GQM no suporte a definição de métricas.** Disponível em: <http://tede.pucrs.br/tde_busca/arquivo.php?codArquivo=2293>. Acesso em: 01 junho 2010.

DEMARCO, Tom. **Controle de projetos de software**. 9.ed. Rio de Janeiro: Campus, 1991.

FIGUEIREDO, Paulo H. **Pesquisa-ação.** Disponível em: <http://www.webartigos.com/articles/21496/1/Pesquisa-acao/pagina1.html#ixzz113d1 FrvF>. Acesso em: 30 setembro 2010.

FILHO, Wilson de Pádua Paula. **Engenharia de software:** fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

GIL, Antonio C. **Como elaborar projetos de pesquisa.** São Paulo: Atlas, 2006.

_____. **Pesquisa Social.** 5ª edição. São Paulo: Atlas, 2007.

HOLANDA, Victor B.; RICCIO, Edson L. **A utilização da pesquisa ação para perceber e implementar sistemas de informações empresariais.** Disponível em: < http://www.ccsa.ufrn.br/depad/simulacaoempresarial/downloads/textos/Utiliza%E7%E3o %20de%20pesquisa-a%E7%E3o.pdf>. Acesso em: 14 outubro 2010.

IEEE standard glossary of software engineering terminology, 1990. IEEE Std 610.12.

ISO9126-2 – **Software engineering – product quality – part2**: External metrics, ISO/IEC, 2003.

ISO9126-3 – **Software engineering – product quality – part3**: Internal metrics, ISO/IEC, 2003.

MACHADO, M. P.; SOUZA, S. F. **Métricas e qualidade de software**. Disponível em <http://www.fattocs.com.br/download/qualidade-sw.pdf>. Acesso em: 20 junho 2010.

MARCONI, Marina A.; LAKATOS, Eva M. **Técnicas de pesquisa.** São Paulo: Atlas, 1999.

OLIVEIRA, Karlos T. M. Vital. **Uma abordagem para promover o alinhamento entre a estratégia de negócio e a tecnologia de informação.** Disponível em <http://www.ppgsc.ufrn.br/html/Producao/Dissertacoes/KarlosThadeuMatiasVitaldeOliveira.pdf>. Acesso 22 junho 2010.

PARK, Robert E.; GOETHERT, Wolfhart B.; FLORAC, William A. **Goal-Driven Software Measurement - a guidebook**. CMU/SEI-96-BH-002, Software Engineering Measurement and Analysis. Carnegie Mellon University, August 1996.

PINTO, Eviston B.: Mapeamento de métricas propostas na série de normas ISO/IEC 9126 para o domínio de educação a distância. Disponível em: < http://monografias.cic.unb.br/dspace/ bitstream/123456789/180/1/MonografiaEviston Licenciatura.pdf>. Acesso em: 10 outubro 2010.

POZZEBON, Marlei.; FREITAS, Henrique M.R. **Pela aplicabilidade**: com um maior rigor científico dos estudos de caso em sistemas de informação. Revista periódica: 1998.

PRESSMAN, Roger S. **Engenharia de software**. São Paulo: Pearson Makron Books, 1995.

_____. **Engenharia de software**. 5ª edição. Rio de Janeiro: McGrall-Hill, 2002.

_____. **Engenharia de software**. 6ª edição. Porto Alegre: McGrall-Hill, 2010.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico.** Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. Novo Hamburgo: Feevale, 2009.

ROESCH, Sylvia M. A. **Projetos de estágio e de pesquisa em administração**. 3ª edição. São Paulo: Atlas, 2006.

SATO, Danilo. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software.** Disponível em: <http://grenoble.ime.usp.br/~gold/orientados/dissertacaoDaniloSato.pdf>. Acesso em: 01 junho 2010.

SODRÉ, Cibele C. P. **Norma ISO/IEC 9126:** Avaliação de qualidade de produtos de software. Disponível em < http://www2.dc.uel.br/nourau/document/?view=625>. Acesso em: 05 junho 2010.

VAZQUEZ, Carlos E.; SIMÕES, Guilherme S.; Albert, Renato M. **Análise de pontos de função**: medição, estimativas e gerenciamento de projetos de software. São Paulo: Érica, 2008.

VERGARA, C. **Projetos e relatórios de pesquisa em administração**. 8ª edição. São Paulo: Atlas, 2007.

ANEXO A – EXEMPLO DE MÉTRICAS EXTERNAS

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Cobertura de implementação funcional	Identificar a quantidade de funções que estão de acordo com a especificação	X = A / B A = número de funções corretamente implementadas, confirmadas através de execução de testes B = número de funções descritas na especificação	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade
Especificação de estabilidade funcional (volatilidade)	Identificar a estabilidade da especificação funcional de um sistema correto depois de entrar em produção	X = 1 - (A / B) A = número de funções mudadas depois de ter sido colocado em operação durante um período específico B = número de funções especificadas	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade

Figura 14 - Métricas externas de adequação
Fonte: Sodré (2006, p.36).

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Acurácia computacional	Qual a freqüência em que o usuário encontra inacurácia em resultados que especificou	X = A / T A = número de inacurácias de usuário encontradas T = Tempo de operação	X >= 0 Quanto mais próximo de 0, melhor.	Absoluta	A=quantidade T = tempo X=quantidade/ tempo

Figura 15 - Métricas externas de acurácia
Fonte: Sodré (2006, p.36).

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Troca de dados (baseado no formato de dado)	Quão completas são as interfaces de função jusantes?	X = A / B A = número de formatos de dados que estão aprovados para ser trocado com outro *software* ou sistema durante teste em dados para troca B = total de formatos de dados para troca	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade
Troca de dados (baseado em tentativas bem sucedidas do usuário)	Quão bem sucedidas são as transferências de dados de informação?	X = 1 - (A / B) A = número de vezes que o usuário não conseguiu trocar tipos de dados com outro *software* ou sistema por motivos de falha B = número de vezes que o usuário tentou trocar dados	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade
Consistência de interface padrão intersistema	O padrão para projeto de interface identificado na especificação segue consistentemente?	X = A / B A = número de "checagem" de itens da interface de intersistema que estão aprovadas por teste, os quais são consistentes com padrões/regras do intersistema B = total de número de "checagem" de itens de interface de intersistema	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade

Figura 16 - Métricas externas de interoperabilidade
Fonte: Sodré (2006, p.37).

ANEXO B – EXEMPLO DE MÉTRICAS INTERNAS

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Cobertura de implementação funcional	Proporção de implementação funcional na revisão	X = A / B A = número de funções implementadas confirmadas na revisão B = número de funções descritas na especificação	0 <= X <= 1 Quanto mais próximo de 1, mais completo.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade
Adequação da implementação funcional	Proporção de adequação da implementação funcional na revisão	X = A / B A = número de funções com problemas que foram detectadas na revisão B = número de funções revisadas	0 <= X <= 1 Quanto mais próximo de 1, mais adequado.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade

Figura 17 - Métricas internas de adequação
Fonte: Sodré (2006, p.38).

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Acurácia computacional	Proporção de figuras significantes que se combinam	X = A / B A = número de itens de dados que implementam figuras significantes especificadas confirmados na revisão B = número de itens de dados que são especificados como figuras significantes	0 <= X <= 1 Quanto mais próximo de 1, mais completo.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade
Precisão da acurácia	Proporção de precisão requerida para implementação	X = A / B A = número de itens de dados que implementam o nível de precisão de acurácia especificada, confirmados pela revisão B = número de itens de dados especificados que requerem precisão de acurácia	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Absoluta	A=quantidade B=quantidade X=quantidade/ quantidade

Figura 18 - Métricas internas de acurácia
Fonte: Sodré (2006, p.39).

Nome da métrica	Propósito da métrica	Medida e fórmula	Interpretação	Tipo de escala	Tipo de medida
Capacidade de troca de dados (formatos básicos de dados)	Proporção de dados em formato compatível	X = A / B A = número de formatos de dados que implementam consistência de formato confirmados pela revisão B = total de formatos de dados para troca	0 <= X <= 1 Quanto mais próximo de 1, mais consistente.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade
Consistência de interfaces (formatos básicos de interfaces)	Proporção de interfaces em formato compatível	X = A / B A = número de formatos de interfaces que implementam consistência de formato confirmados pela revisão B = número de formatos de interface para troca	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade
Consistência de padrões intersistema	Proporção de padrões compatíveis	X = A / B A = número de itens que implementam consistência com regras e padrões confirmados pela revisão B = total de itens que são consistentes com regras e padrões	0 <= X <= 1 Quanto mais próximo de 1, melhor.	Taxa	A=quantidade B=quantidade X=quantidade/ quantidade

Figura 19 - Métricas internas de interoperabilidade
Fonte: Sodré (2006, p.40).

ANEXO C – ANÁLISE DE CONTEÚDO DA CATEGORIA MÉTRICAS

Categoria Inicial (Perguntas[12])	Categoria Intermediária (Respostas)	Categoria Final (Categorias Inferidas)
1) Não se pode controlar o que não se mede (DEMARCO, 1982). Você considera esta afirmação válida quando o assunto é desenvolvimento de software? Justifique.	- *Gerente de TI*: "Sim. Sem medir, não se pode saber de forma quantitativa, o estado atual de um processo. Fica-se somente com o "sentimento" de atrasado ou de "no prazo" e isto pode variar de pessoa para pessoa". - *Analistas de Negócios:* *I)* "Sim, é impossível avaliar/controlar uma equipe de desenvolvimento de forma eficiente e eficaz sem ter números para determinar os critérios esperados durante o desenvolvimento do software. Através de medições também será possível iniciar um processo de melhoria continua através da gestão desses números". *II*) "Sim, não se consegue ter uma idéia do problema e nem da solução necessária, sem medir". *III)* "Sim considero válida esta informação pois não temos como controlar o desenvolvimento de um software se não tivermos a medida do que se vai desenvolver em numero de processos e do tempo previsto para cada desenvolvimento". - *Analista de sistemas:* "Sim, considero muito importante a utilização de métricas para avaliar o processo de desenvolvimento de software e também o produto desenvolvido. A utilização de métricas permite quantificar a qualidade e produtividade do processo de desenvolvimento, e desta forma estabelecer parâmetros para avaliação e controle deste processo".	- Controle; - Processo desenvolvimento, e - Melhoria contínua.
2) Você considera importante utilizar técnicas para medir o esforço no desenvolvimento de software? Por quê?	- *Gerente de TI:* "Sim, porque a estimativa dos prazos para conclusão de atividades de desenvolvimento de software continua tendo um baixo nível de acerto e para aumentar a precisão destas estimativas é fundamental que se consiga medir o que é feito para saber como estimar. - *Analistas de Negócios:* *I)* "Sim, pois será possível estimar prazos mais assertivos evitando desgastes com os clientes (internos ou externos) e também para facilitar o acompanhamento do tempo realizado versos o orçado". *II*) "Sim, dá mais confiabilidade aos prazos de desenvolvimento e aos prazos do projeto". *III*) "Sim, por que utilizando técnicas para medir o esforço no desenvolvimento de software temos como controlar o prazo do desenvolvimento, administrar os recursos e garantir um acerto de praticamente 100% no comprometimento da entrega do projeto". - *Analista de sistemas:* "Considero importante, pois o esforço é um fator determinante do custo e tempo do processo de desenvolvimento. Medir o esforço permite criar uma base de dados e experiência para futuramente serem utilizadas em estimativas e avaliação do esforço em novos desenvolvimentos".	- Estimativa de esforço; - Prazos, e - Acertividade.
3) Principais exemplos de medidas de software segundo Sommerville são: medidas relacionadas a pontos de função (considera as funcionalidades a ser entregue ao usuário) e linhas de códigos (processo de contar linhas de programação de um programa fonte). Qual tipo de medida que você considera mais	- *Gerente de TI*: "Considero que pontos de função é mais adequado, pois mesmo sabendo que na análise caso-acaso a probabilidade de erro é grande, na média, a técnica produz resultados bons e, principalmente, de forma rápida. Já a medição por linha de código não me parece fazer mais sentido nos dias de hoje com a disseminação das ferramentas 'case' de desenvolvimento". - *Analistas de Negócios:* *I*) "Penso ser mais adequada a medida de linhas de código, por ser objetiva e tangível, já o ponto de função é um pouco intangível abrindo margem para discussões além de necessitar de uma pessoa experiente e	- Ponto de função; - Bons resultados , e clareza no entendimento do requisito.

[12] A numeração indicada em cada pergunta identifica o número da questão no questionário utilizado.

adequado a ser utilizado na atualidade? Justifique.	conhecedora do assunto, ao contrario de uma contagem de linhas de código". *II*) "Ponto de função, é o que conheço, quando bem aplicado e bem exercitado, torna possível a medição do esforço". *III*) "Pra mim a medida mais adequada é o Ponto de Função, pois este tipo de medida permite que o usuário e a TI conversem em uma linguagem entendível por ambos, ou seja, o usuário visualiza o que será desenvolvido para atender a suas necessidades e o desenvolvedor sabe o que deve ser realizado de forma clara facilitando a previsão do tempo de desenvolvimento". - *Analista de sistemas:* "Considero a medida por pontos de função mais adequada, pois esta é mais fácil de ser elaborada com base nos requisitos de desenvolvimento ou manutenção do software. Também é mais fácil de ser medida e avaliada quanto a complexidade e esforço. A contagem de linhas de código muitas vezes é impraticável nos ambientes atuais de desenvolvimento onde as ferramentas, ou frameworks de desenvolvimento, trabalham com o código inserido em contextos (ex: ação de um botão), ou geram códigos automáticos".	
10) Segundo Koscianski (2007), ao definir uma determinada métrica, deve ser identificado primeiramente qual objetivo será alcançado. Desta mesma forma o processo de medição do MPSBR (nível F) sugere utilizar métodos que definem métricas orientadas a objetivos (SOFTEX, 2009). Você considera esta sugestão válida? Por quê?	- *Gerente de TI*: "Sim, pois para medir alguma coisa é necessário saber quando o mesmo esta concluído, ou seja, que o objetivo foi alcançado". - *Analistas de Negócios:* "Sim, pois considero que quando se tem um objetivo vinculado a uma métrica, torna o processo mais tangível facilitando o atendimento das métricas estipuladas". *II) "*Não conheço". *III*) "Sim, por que eu não teria como medir o esforço sem saber o objetivo e o que deverá ser desenvolvido. - *Analista de sistemas:*"Considero válido, pois acredito que métricas orientadas a objetivos são mais fáceis de serem definidas , avaliadas e tem grande valia no processo de medição".	- Métricas focada no objetivo, e - Facilidade para definir métricas.
14) Na opinião de Pressman (2002) percebe-se que, a cultura de medição de software não é comum em pequenas empresas de desenvolvimento de software. Em sua opinião, cite 2 (duas) razões porquê isto acontece.	- *Gerente de TI*: "1) medir software exige tempo para aplicação que a maioria das pequenas empresas não está disposta a usar; 2) medir software necessita de histórico de desenvolvimento que estas empresas não têm". - *Analistas de Negócios:* *I) "*Quadro de pessoal reduzido, em muitos casos os próprios donos trabalham nas áreas técnicas desempenhando diversas funções sem terem tempo para se dedicarem a implantação de uma cultura de medição; Mão de obra pouco qualificada, que em muitos casos se sentem ameaçados pelas métricas"; *II) "* Falta de conhecimento do empreendedor e recursos financeiros para realização". *III) "*Falta de processo interno; falta de conhecimento da cultura de medição". - *Analista de sistemas:* *"D*esconhecimento desta prática e da forma de executá-la. Não considera importante ou acha que não dá retorno a medição de software.	- Falta de tempo; - Desconhecimento da técnica, e - Procedimentos internos.
19) Atualmente você considera utilizar, usufruir de métricas que forneçam medidas de esforço e qualidade nos softwares desenvolvidos?	- *Gerente de TI*: "Sim". - *Analistas de Negócios:* *I) "Não".* *II) "Não".* *III) "Não".* - *Analista de sistemas:* "Considero a utilização, pois acho que é importante".	- Não é utilizado.
22) Segundo PRESSMAN	- *Gerente de TI*:	- Recurso

(2002) muitas empresas de desenvolvimento não possuem programas abrangentes de métricas de software. Em sua opinião, quais motivos levam as empresas a NÂO investirem em programas de medição de software?	"Porque medir tem um custo e como os benefícios da medição não são percebidos pelos clientes, a maioria das empresas não investe". - *Analistas de Negócios:* I) "Imagino que pelo pouco conhecimento do assunto e pela falta de tempo para pesquisar, só percebem pontos negativos como maior tempo na entrega, maior custo de desenvolvimento, etc, e por isso acabam não investindo". *II*) "Não conhecimento das técnicas e falta de recursos financeiros". *III*) "Falta de conhecimento; Falta de processo interno; Falta de preocupação com a qualidade do software e foco no prazo de entrega sem se preocupar com o que esta sendo entregue".	financeiro; - Falta percepção da importância para o produto, e - Falta de conhecimento.

ANEXO D – ANÁLISE DE CONTEÚDO DA CATEGORIA PROCESSO DE DESENVOLVIMENTO

Categoria Inicial (Perguntas)	Categoria Intermediária (Respostas)	Categoria Final (Categorias Inferidas)
7) Empresas de desenvolvimento de software buscam certificações em CMMI, MPSBR. Você considera que existe melhora na qualidade do desenvolvimento utilizando os métodos aplicados por estas certificações? Se SIM, cite 3 (três) vantagens da sua utilização; caso contrário, justifique.	- *Gerente de TI*: "Não considero que por si só as certificações melhorem a qualidade de desenvolvimento de software. Isto não é garantido por estas certificações, elas garantem que pelo menos um projeto foi feito por um processo documentado". - *Analistas de Negócios:* *I) "*Sim, penso que a utilização desses métodos trazem benefícios como: 1)Instituição de um processo para desenvolvimento de software; 2) Definição métricas de qualidade; 3) Maior assertividade na definição dos prazos de desenvolvimento". *II) "*Sim, São metodologias testadas por muitas empresas. Vantagens: Padronização, controle e qualidade". *III) "*Sim, tendo como vantagens: 1) Processos são definidos e documentados; 2) Os papéis e responsabilidades de cada um na equipe ficam bem definidos; 3) Pode se prever prazos, qualidade e custos com um nível de acerto em quase 100%". - *Analista de sistemas:* *"*Considero que existe melhora na qualidade do desenvolvimento: 1) Melhora na qualidade do processo de desenvolvimento; 2) Melhora na qualidade do produto desenvolvido; 3) Produto com maior valor agregado em função da melhora do processo e da qualidade".	-Processo de desenvolvimento; - Padronização; - Qualidade, e documentação.

ANEXO E – ANÁLISE DE CONTEÚDO DA CATEGORIA MELHORIA CONTÍNUA

Categoria Inicial (Perguntas)	Categoria Intermediária (Respostas)	Categoria Final (Categorias Inferidas)
12) Medições fornecem um feedback (retorno) para melhoria contínua do processo (PETERS, 2001). Você considera importante utilizar uma ferramenta que traz melhoria para processo de desenvolvimento? Por quê?	- *Gerente de TI*: "Sim, porque assim o processo de desenvolvimento é facilmente controlado". - *Analistas de Negócios:* *I) "*Sim, considerando a grande velocidade em que as mudanças ocorrem no mundo corporativo atrelado a necessidade de respostas rápidas as mudanças do mercado, considero muito importante a utilização de ferramentas que dêem suporte a área de desenvolvimento de software, permitindo assim o processo de melhoria continua ganhando cada vez mais a confiança da área usuária". *II*) *"*Sim, permite aprimorar os conceitos e técnicas de desenvolvimento". *III*) *"*Sim, por que para se ter qualidade de software um dos requisitos é a previsão de melhoria e isto se fazendo com ferramentas adequadas traz ganho ao processo". - *Analista de sistemas:* "Considero importante, pois a avaliação do retorno gera experiência, que é um fator importante para fazer estimativas de esforço, análise de complexidade das funções e várias outras vantagens".	- Controle; - Busca pela melhoria, e - Qualidade para o processo de desenvolvimento.

ANEXO F – ANÁLISE DE CONTEÚDO DA CATEGORIA QUALIDADE

Categoria Inicial (Perguntas)	Categoria Intermediária (Respostas)	Categoria Final (Categorias Inferidas)
11) Em sua opinião, o que significa qualidade de software?	- *Gerente de TI*: "Software desenvolvido através de uma metodologia que documente e permita o rastreamento, que seja entregue dentro do prazo acordado e que atenda as especificações". - *Analistas de Negócios:* *I*) "É um software que atenda a necessidade do usuário final quanto a usabilidade, confidencialidade, integridade dos dados e um tempo de resposta menor possível considerando a sua complexidade de processamento relacionamento com outros sistemas". *II*) " Um produto que cumpra o seu papel, atenda as necessidades do cliente, que transmita confiabilidade e de fácil manutenção". *III*) "Qualidade de Software significa processos bem definidos, documentados e utilizados; prever prazos, custos e qualidade com segurança; definir requisitos e medir esforço com precisão; prever melhorias continuas e buscar um alto grau de satisfação do usuário". - *Analista de sistemas:* "Significa aplicar métodos para que o software atenda os seus requisitos funcionais e não funcionais. Software funcional, sem erros, com precisão nos cálculos, velocidade de execução aceitável e executando na plataforma definida".	- Prazo acordado; - Entrega dos requisitos solicitados; - Confiabilidade, e - Documentação.
17) A norma ISO/IEC 9126 trata de um modelo de qualidade para um produto de software, que permite orientar diferentes perspectivas de avaliação. Por exemplo: desenvolvedores e clientes tem visões e preocupações diferentes relacionadas a qualidade do mesmo produto (KOSCIANSKI, 2007, p.206). Cite 2 (dois) motivos que possam justificar a afirmação acima.	- *Gerente de TI*: "1) Desenvolvedores querem entregar o que entenderam dos requisitos; 2) clientes esperam receber o que sonham e não o que especificam". - *Analistas de Negócios:* *I)* "1) Desenvolvedores estão muito focados na qualidade técnica do desenvolvimento (padrões, métodos, etc), já os clientes estão preocupados com a usabilidade; 2) Desenvolvedores focam na qualidade de uma tela especifica, já o cliente tem uma preocupação com a visão sistêmica, criticando todas as conexões existentes entre os dados". *II)* "Não conheço". *III)* "1) Usuário esta preocupado com a facilidade do uso do sistema e se o mesmo atende a suas necessidades; 2) Desenvolvedor esta preocupado com a segurança dos dados fornecidos, cumprimento dos prazos, adequação das funcionalidades para atender o usuário, ou seja em desenvolver com qualidade dentro do tempo previsto". - *Analista de sistemas:* "1) Clientes tem uma visão mais funcional do software do que o desenvolvedor, preocupa-se que o software seja "prático", rápido e funcional, enquanto que o desenvolvedor preocupa-se com funcionalidades e atendimento dos requisitos, e muitas vezes o entendimento do Requisito de Sistema é diferente entre o usuário e o desenvolvedor; 2) O cliente não tem conhecimento técnico para avaliar considerar determinados quesitos de qualidade que se referem ao desenvolvimento. Qualidade do código, do modelo implementado e outros".	- Desenvolvimento do software; - Funcionalidade; - Facilidade de uso, e - Entendimento dos requisitos.

ANEXO G – QUESTIONÁRIO

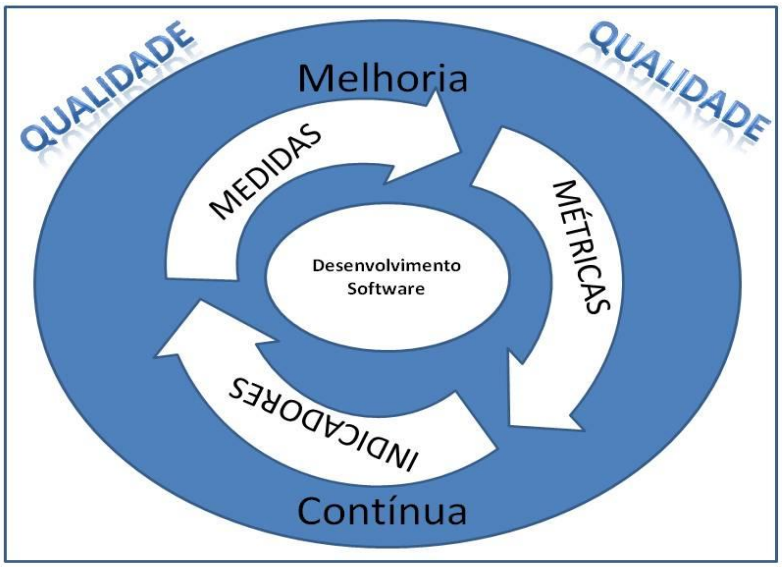
QUESTIONÁRIO DE VALIDAÇÃO - TRABALHO DE CONCLUSÃO

JOCEMAR SCHMITT MARIA

Universidade Feevale
Instituto de Ciência Exatas e Tecnologia
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso

Professor orientador: Ms Roberto Scheid

Novo Hamburgo, Outubro de 2010.



Quando puder medir aquilo de que está falando e expressar em números, você saberá algo a respeito. (Lord Kelvin).

Não se pode controlar o que não se pode medir (DEMARCO, 1982).

Ao passar do tempo o software ganhou mais importância do que o hardware, o software ao contrário do hardware frequentemente é o item de maior custo (PRESSMAN,

1995). Devido a tal importância dada ao software faz-se necessário uma preocupação maior, e a engenharia de software veio para ajudar neste quesito.

O questionário a seguir faz parte de uma pesquisa-ação que está sendo elaborada com objetivo de trazer melhoria para um processo onde estimar a produção de software com qualidade é um dilema. Para um melhor entendimento, é indispensável uma explicação dos principais conceitos aqui listados.

Medição de software – tornar algo intangível em quantidade numérica. As medições fornecem um feedback valioso e ajudam no controle e melhoria do processo, pois quantificam a avaliação de um produto de software (PETERS, 2001).

Processo de desenvolvimento – Sequência de atividades efetuadas na elaboração do software (PETERS, 2001). Para Pressman, o processo de desenvolvimento deve preocupar-se com a qualidade e prazos, e define processo de software como sendo uma série de passos previsíveis – um roteiro que ajuda a criar um resultado de alta qualidade.

Qualidade – A qualidade é relativa. O que é qualidade para uma pessoa pode ser falta de qualidade para outra (G. Weinberg). Weinberg apud Koscianski (2007, p.26) complementa ainda "os requisitos foram definidos por alguém, logo a qualidade depende das escolhas que alguém efetuou". Segundo Sommerville (2003), "Diferentes *stakeholders* têm em mente diferentes requisitos e podem expressá-los de maneiras distintas. Os engenheiros de requisitos precisam descobrir todas as possíveis fontes de requisitos e encontrar pontos comuns e os conflitos". Considerando qualidade da forma exposta acima se entende então que para ter qualidade necessita-se ter objetivos definidos.

Melhoria contínua – Buscar a melhoria do processo através da identificação de itens a melhorar, colocar em prática as soluções selecionadas e aplicar métodos de controle.

Muito obrigado pela colaboração!

1) Não se pode controlar o que não se mede (DEMARCO, 1982). Você considera esta afirmação válida quando o assunto é desenvolvimento de software? Justifique.

2) Você considera importante utilizar técnicas para medir o esforço no desenvolvimento de software? Por quê?

3) Principais exemplos de medidas de software segundo Sommerville são: medidas relacionadas a pontos de função (considera as funcionalidades a ser entregue ao usuário) e linhas de códigos (processo de contar linhas de programação de um programa fonte). Qual tipo de medida que você considera mais adequado a ser utilizado na atualidade? Justifique.

4) Marque abaixo o(s) item(s) que você considera importante quando se fala em utilizar técnicas de medição de software em um processo de desenvolvimento.
   ( ) Torna o processo mais demorado sem trazer benefícios.
   ( ) Aumenta o tempo de entrega do produto, porém é importante.
   ( ) Agrega valor ao produto.
   ( ) Irrelevante a sua utilização.

5) À medida que o software ganha espaço, a engenharia de software preocupa-se com a qualidade do produto. Em relação à métricas de qualidade, na lista abaixo marque uma alternativa que você mais considera:
   ( ) Um modismo.
   ( ) Importante no processo de desenvolvimento.
   ( ) Melhora o processo interno.
   ( ) Necessário para ter um produto de qualidade.
   ( ) Outra. Qual?_____.

6) Considerando as definições apresentadas a respeito de medição de software, qual das alternativas a seguir você considera agregar mais valor:
   ( ) Ao cliente.
   ( ) A área de sistemas.
   ( ) Ao produto.
   ( ) Ao processo de desenvolvimento como um todo.
   ( ) Nenhuma das alternativas.

7) Empresas de desenvolvimento de software buscam certificações em CMMI[13], MPSBR[14]. Você considera que existe melhora na qualidade do desenvolvimento utilizando os métodos aplicados por estas certificações? Se SIM, cite 3 (três) vantagens da sua utilização; caso contrário, justifique.

8) Conforme abordado na primeira página desta pesquisa, qualidade depende de uma definição clara do que se espera. Qual das opções a baixo é mais indicada para medir qualidade de software.
   ( ) Métricas que qualificam o software como produto.
   ( ) Feedback do cliente – usuário.
   ( ) Tempo de utilização do software.

[13] CMMI (Capability Maturity Model for Software) – Modelo de capacitação de processo patrocinado pelo Departamento de Defesa dos EUA, para avaliação da capacidade dos seus fornecedores de software KOSCIANSKI (2007).
[14] MPSBR (Melhoria do Processo do Software Brasileiro) – Este modelo atende a necessidade de implantar os princípios de engenharia de software de forma adequada ao contexto das empresas brasileiras, seguindo as abordagens internacionais para definição, avaliação e melhoria do processo de software KOSCIANSKI (2007).

( ) Não tenho opinião formada.
( ) Outra:Qual? _____.

9) Se o resultado de uma medida é utilizado para julgar um software, é preciso ter certeza que ele está sendo feito da maneira correta (KOSCIANSKI, 2007). Dentre as alternativas a baixo marque a(s) alternativa(s) que, em sua opinião, podem ser usadas como forma de medir software.
( ) Volume de dados (software suporta volume grande de dados).
( ) Qualidade dos dados medidos (informações confiáveis).
( ) Precisão das medidas.
( ) Experiência do medidor.
( ) Outra:Qual? _____.

10) Segundo Koscianski (2007), ao definir uma determinada métrica, deve ser identificado primeiramente qual objetivo será alcançado. Desta mesma forma o processo de medição do MPSBR (nível F) sugere utilizar métodos que definem métricas orientadas a objetivos (SOFTEX, 2009). Você considera esta sugestão válida? Por quê?

11) Em sua opinião, o que significa qualidade de software?

12) Medições fornecem um feedback (retorno) para melhoria contínua do processo (PETERS, 2001). Você considera importante utilizar uma ferramenta que traz melhoria para processo de desenvolvimento? Por quê?

13) Classifique de 1 a 5 os itens abaixo por ordem de prioridade, quanto menor o número maior a importância do item, que provêem contribuição ao processo de desenvolvimento de software?
( ) Processo de melhoria contínua para definição de métricas de qualidade.
( ) Medição de software.
( ) Certificações (CMMI, MPSBR).
( ) Definição e aprovação de requisitos.
( ) Outra: Qual? _____.

14) Na opinião de Pressman (2002) percebe-se que, a cultura de medição de software não é comum em pequenas empresas de desenvolvimento de software. Em sua opinião, cite 2 (duas) razões porquê isto acontece.

15) Na sua opinião, além da construção de um sistema, quais atividades a baixo NÂO deveriam deixar de fazer parte do processo de desenvolvimento de software?
( ) Levantamento de requisitos.
( ) Testes unitários.
( ) Testes integrados.
( ) Estimativa de esforço.
( ) Documentação.
( ) Homologação.
( ) Definição do prazo de entrega.

16) Para se obter uma medida de software existem diversas métricas. Marque na lista a seguir qual(is) você já ouviu falar ou teve algum contato.

( ) LOC – Linhas de código.
( ) APF – Análise de pontos de função.
( ) COCOMO – Construtive Cost Model.
( ) UCP – Use Case Points.
( ) Nenhuma a cima.

17) A norma ISO/IEC 9126 trata de um modelo de qualidade para um produto de software, que permite orientar diferentes perspectivas de avaliação. Por exemplo: desenvolvedores e clientes tem visões e preocupações diferentes relacionadas a qualidade do mesmo produto (KOSCIANSKI, 2007, p.206). Cite 2 (dois) motivos que possam justificar a afirmação acima.

18) Um modelo de qualidade foi definido pela norma ISO/IEC 9126 (abaixo segue as características desse modelo). Primeiramente ler todas, após priorizar de 1 (mais importante) a 6 (menos importante) as características considerando o seu grau de importância na visão do desenvolvedor de sistemas.
( ) Funcionalidade – diz respeito aquilo que o software faz quando solicitado pelo usuário, podemos dizer que são os requisitos funcionais.
( ) Manutenibilidade – está relacionada a facilidade de modificação de um software, não deve ser confundida com a capacidade de configuração do software.
( ) Usabilidade – a usabilidade representa basicamente o quão fácil é de usar o produto.
( ) Confiabilidade – a confiabilidade de um programa se traduz como a capacidade de manter um certo nível de desempenho quando operando em um certo contexto de uso.
( ) Eficiência – a norma ISO/IEC 9126 estabelece duas dimensões para avaliar a eficiência: o tempo (velocidade de execução do produto) e a utilização dos recursos (engloba todo o restante que não seja o tempo de CPU: quantidade de memória, carga de CPU, ocupação de disco etc).
( ) Portabilidade – abrange a idéia de portar aplicações entre organizações diferentes. Em tese, supõe-se que um programa passa, então, ser elaborado para operar em ambientes com características diferentes.

19) Atualmente você considera utilizar, usufruir de métricas que forneçam medidas de esforço e qualidade nos softwares desenvolvidos?

20) Quanto à funcionalidade do software - considerando o seu dia-a-dia, o processo de desenvolvimento existente, marque a(s) questão(s) abaixo, que você considera que deveria existir métricas (caso não tenha). O software:
( ) Propõe-se a fazer o que é apropriado?
( ) Gera resultados corretos ou conforme acordados?
( ) É capaz de interagir com os sistemas especificados?
( ) Evita acesso não autorizado, acidental ou deliberado a programas e dados?
( ) Está de acordo com normas e convenções previstas em leis?
( ) Outra:Qual? _____.

21) Em sua opinião, marque a(s) questão(s) que considera importante medir quando relacionado a qualidade de software.
( ) Com que frequência apresenta falhas;
( ) É capaz de recuperar dados após uma falha;

( ) É fácil aprender a usar;
( ) É fácil encontrar uma falha quando ocorre;
( ) Há grandes riscos de *bug* quando se faz uma alteração;
( ) É fácil testar quando se faz uma alteração;
( ) Outra: Qual? _____.

22) Segundo PRESSMAN (2002), muitas empresas de desenvolvimento não possuem programas abrangentes de métricas de software. Em sua opinião, quais motivos levam as empresas a NÃO investirem em programas de medição de software?