MBA EM ENGENHARIA DE SOFTWARE

Engenharia de Software com SWEBOK e Métricas

APOSTILA

Versão. 1.0

Índice

Lista de Figuras

Lista de Tabelas

1. APRESENTAÇÃO

Esta apostila foi elaborada como material introdutório de referência para o curso de engenharia de software e métricas.

Aqui serão abordados os principais tópicos e a organização do guia SWEBOK[1] (*Software Engineering Body of Knowledge*) e os conceitos envolvidos nas métricas de software, tanto para as métricas de produto, quanto para as métricas de processo.

[1] No capítulo destinado ao SWEBOK, são apresentados diversos trechos traduzidos do guia original. O SWEBOK é marca registrada do IEEE.

2. SWEBOK

2.1. Introdução e Organização do Modelo

O SWEBOK (*Software Engineering Body of Knowledge*) é um guia que pretende prover uma caracterização validada e consensual dos limites da disciplina da engenharia de software, fornecendo acesso ao corpo de conhecimento desta disciplina.

De maneira mais detalhada, o SWEBOK apresenta 5 (cinco) objetivos, que são:

1. Promover uma visão consistente da engenharia de software mundialmente;
2. Clarear a localização e determinar os limites, da engenharia de software com outras disciplinas como: ciência da computação, gerenciamento de projeto, engenharia da computação e matemática;
3. Caracterizar o conteúdo de uma disciplina de engenharia de software;
4. Prover um acesso no formato de lista do corpo de conhecimento em engenharia de software;
5. Prover os fundamentos para o desenvolvimento de um curriculum, ou certificações individuais ou material de licenciamento.

Este corpo de conhecimento está organizado no guia em áreas de conhecimento, sendo 10 relativas à engenharia de software e 1 de disciplinas correlatas (Figura 1 e Figura 2).

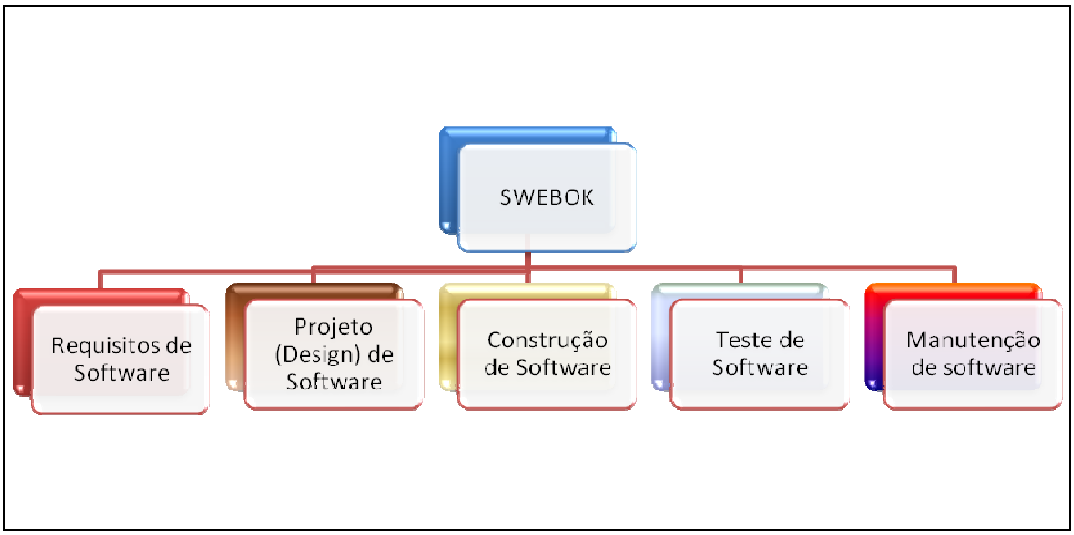

Figura 1 – 5 Primeiras áreas de conhecimento do SWEBOK (SWEBOK, 2004).

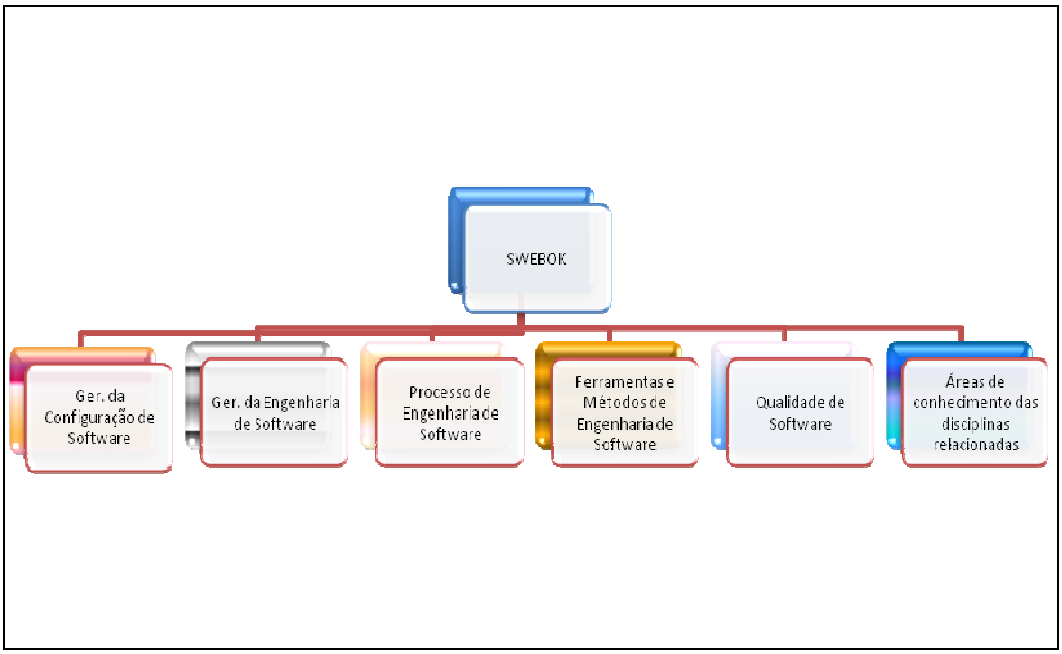

Figura 2 - 6 últimas áreas de conhecimento do SWEBOK (SWEBOK, 2004).

Cada área de conhecimento apresenta uma derivação de tópicos de conhecimento e para cada um destes tópicos são apresentadas descrições dos objetivos e as devidas referências bibliográficas utilizadas (Figura 3).

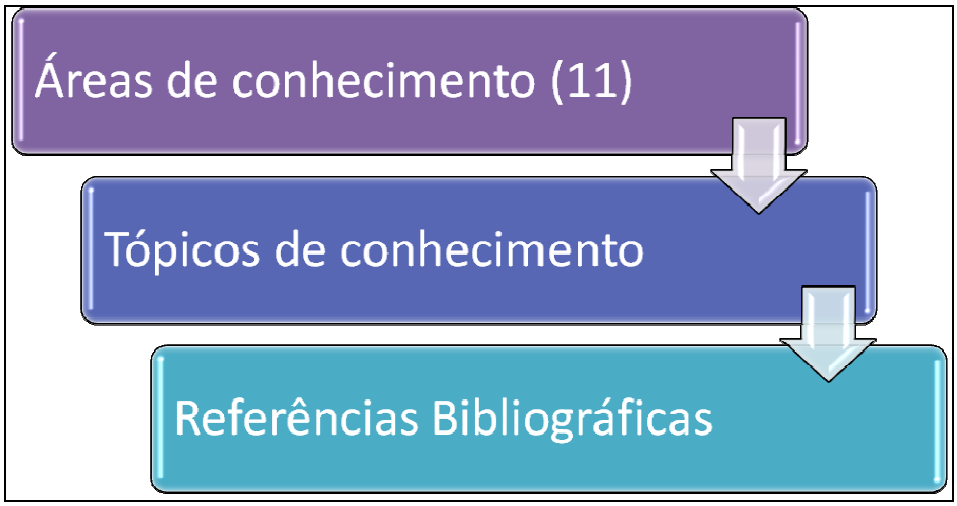

Figura 3 – Esquema da organização do guia SWEBOK.

Por ser um guia genérico sobre as atividades e processos envolvidos na engenharia de software, o SWEBOK não traz nenhum conhecimento relacionado à:

- Linguagens de programação;
- Banco de dados;
- Rede de computadores.

Devido a estas limitações e a sua forte orientação acadêmica a utilização do guia é um pouco restrita, mas pode ser um bom começo na hora de procurar referências para implementar processos dentro de uma organização.

2.2. Áreas de Conhecimento

2.2.1. Requisitos de Software

Segundo o SWEBOK (2004, p. 33), “Um requisito de software é uma propriedade que precisa ser exibida com o objetivo de resolver um problema do mundo real” .

Assim, a disciplina que lida com este tipo de requisito tem por objetivo tratar a elicitação, analise, especificação e validação destes.

Para não fazer confusão com o próprio termo engenharia de software, os autores preferiram não utilizar no guia o termo “Engenharia de Requisitos”.

A disciplina de requisitos de software está subdividida no SWEBOK em 7 tópicos, conforme exibidos na Figura 4.

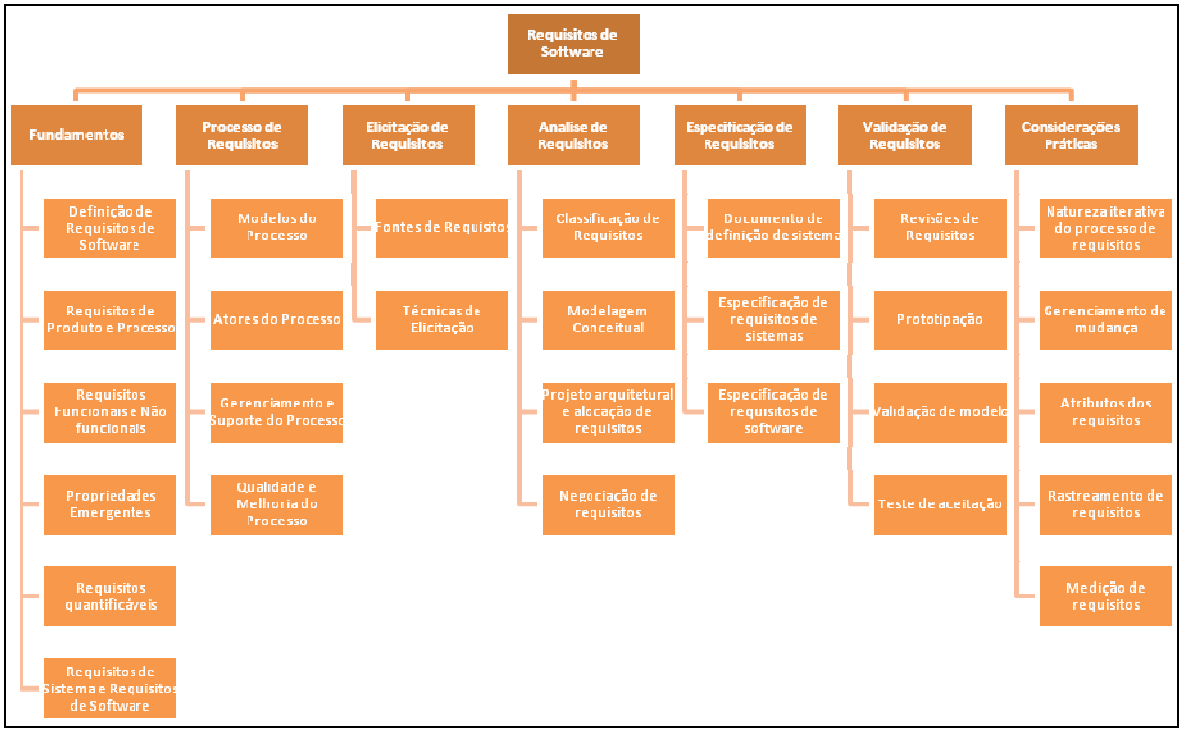

Figura 4 - Tópicos da área de conhecimento Requisitos de Software (SWEBOK, 2004)

Fundamentos

Inclui a definição dos requisitos de software e também seus principais tipos: produtos vs processos, funcional vs não funcional, propriedades emergentes. Também descreve a importância de se quantificar os requisitos e separá-los em requisitos de sistema e requisitos de software.

Processo de Requisitos

Introduz o processo de requisitos, servindo de orientação para as demais subáreas e descreve como a engenharia de requisitos se relaciona com as demais áreas da engenharia de software. Descreve os modelos do processo, atores, o suporte e gerenciamento, e a qualidade e melhoria.

Elicitação de Requisitos

Esta subárea se preocupa em identificar de onde os requisitos são coletados, e como estes podem ser coletados. Inclui fontes de requisitos e técnicas de elicitação.

Análise de Requisitos

Está focada no processo de análise de requisitos para:

- Detectar e resolver os conflitos entre os requisitos.
- Descobrir os limites do software e como este deve interagir com o ambiente.
- Elaborar requisitos de sistema para os requisitos de software.

A análise de requisitos inclui a classificação, modelagem conceitual, projeto arquitetural, alocação e negociação.

Especificação de Requisitos

Trata da produção de um documento (ou versão eletrônica equivalente), que possa ser sistematicamente revisado, avaliado e aprovado.

Validação de Requisitos

Trata do processo de examinar os documentos de requisitos para garantir que eles estão definindo o sistema correto. Está subdividida em descrições dos requisitos, revisões, prototipação, e modelo de validação e teste de aceitação.

Considerações Práticas

Apresenta tópicos que ajudam a entender o processo de requisitos como: a natureza interativa, o gerenciamento de mudança, manutenção dos requisitos, medições, entre outros.

2.2.2. Projeto de software

Segundo o SWEBOK (2004, p. 26), o projeto de software é "o processo de definição de arquitetura, componentes, interfaces e outras características de um sistema ou componente".

Esta área de conhecimento foi organizada em 6 subáreas, conforme apresentado na Figura 5.

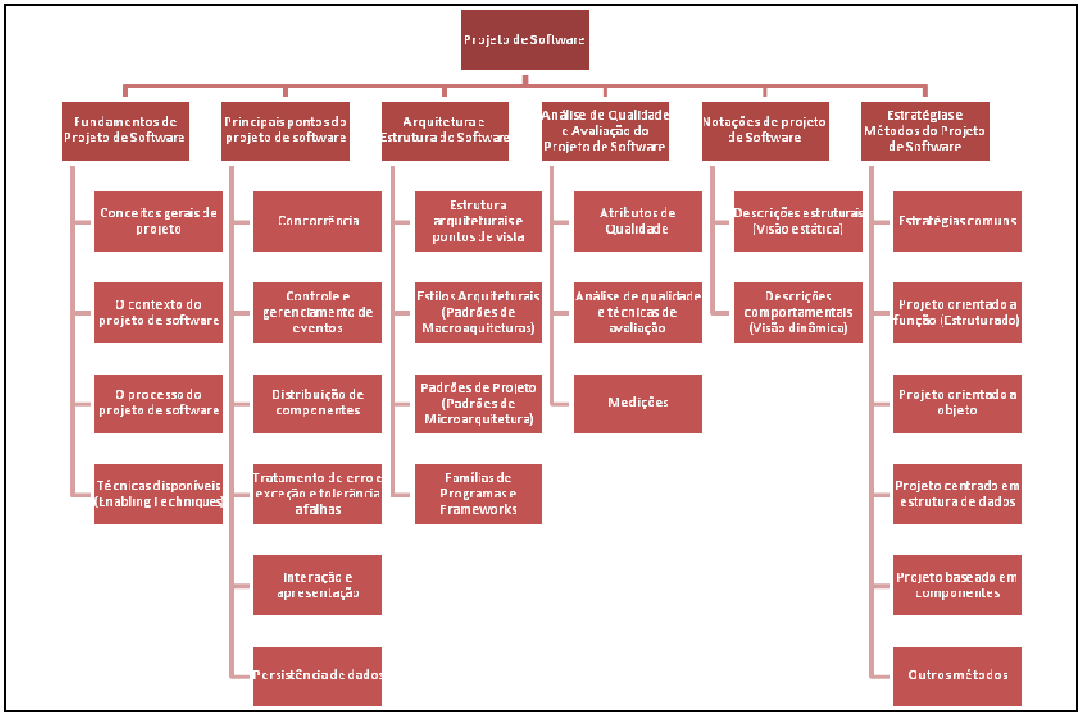

Figura 5 - Tópicos da área de conhecimento Projetos de Software (SWEBOK 2004).

Fundamento do projeto de software

Apresenta a base para o entendimento do papel e escopo do projeto de software. São conceitos gerais de software do projeto de software, processos e técnicas.

Pontos chaves do projeto de software

Inclui concorrência controle e tratamento de eventos, distribuição de componentes, tratamento de erro e exceção, tolerância a falhas, interação e apresentação e persistência de dados.

Arquitetura e Estrutura de Software

Descreve quais são as estruturas arquiteturais, estilos, padrões de projeto, e famílias de programas e frameworks.

Análise de Qualidade e Avaliação do Projeto de Software

Apresenta aspectos de qualidade (atributos, análise, avaliação e técnicas), no contexto do projeto de software.

Notação de projeto de software

Descreve algumas notações e linguagem que ajudam a representar os artefatos de projeto de software. (Ex.: Diagramas de atividade, diagrama entidade relacionamento, diagramas de colaboração, etc.)

Estratégia e métodos do projeto de software

Descreve estratégias gerais seguidas de projeto orientado a função, projeto orientado a objeto, projeto centrado em estrutura de dados; projeto baseado em componentes e outros.

2.2.3. Construção de Software

A construção de software se refere a criação detalhada de um software funcional através da combinação de codificação, verificação, teste unitário, teste de integração e debugging (SWEBOK, 2004). Esta área de conhecimento foi dividida em três subáreas (Figura 6).

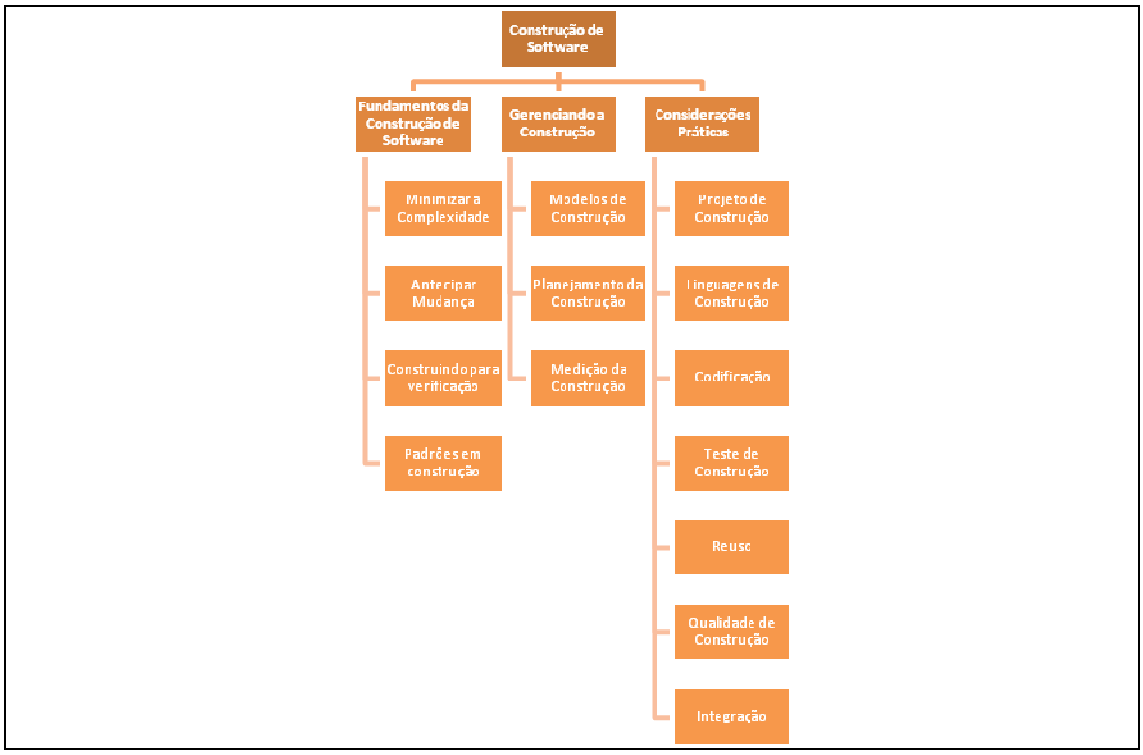

Figura 6 - Tópicos da área de conhecimento Construção de Software (SWEBOK 2004).

Fundamentos da construção de software

Discute alguns dos princípios básicos da construção: minimizar complexidade, antecipar mudança e construção para verificação, além de alguns padrões para construção.

Gerenciar construção

Descreve algumas formas de modelos, planejamento e medições relacionadas à construção de software.

Considerações práticas

Detalha alguns pontos relacionados à linguagens de programação, codificação, teste, reuso e integração.

2.2.4. Teste de software

O teste de software consiste de verificação dinâmica do comportamento de um programa em um conjunto finito de casos de teste, selecionados de um conjunto infinito de execuções (SWEBOK, 2004). Esta área de conhecimento está organizada em 5 subáreas no guia (Figura 7).

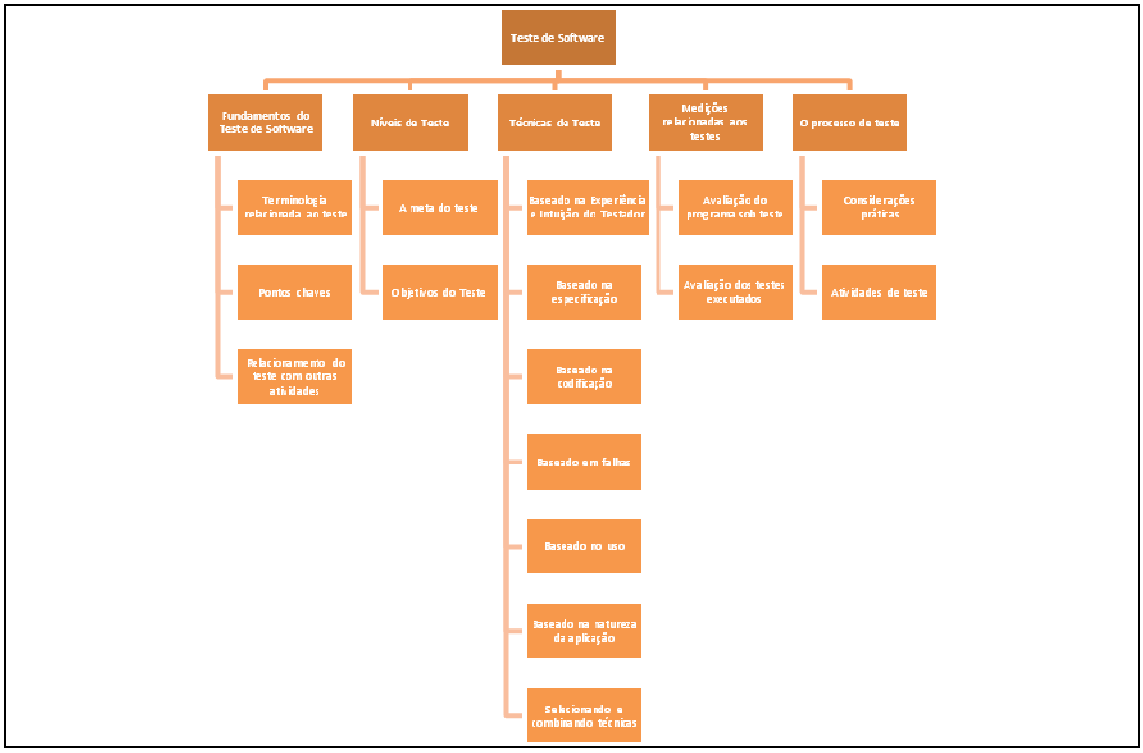

Figura 7 - Tópicos da área de conhecimento Teste de Software (SWEBOK 2004).

Fundamentos do teste de software

Apresenta a terminologia relacionada ao teste de software, os pontos chave, e relacionamento do teste com outras atividades.

Níveis de teste

Descreve os diversos níveis de teste como teste unitário, de integração, de sistema, etc.

Técnicas de teste

Descreve as várias técnicas de teste agrupadas em categorias, como teste baseado na intuição e experiência, baseadas em especificação e em códigos. Apresenta também uma discussão de como selecionar e combinar as técnicas mais adequadas ao projeto.

Medições relacionadas aos testes

Apresenta comentários sobre a medição do teste de software, relacionado ao programa sob teste. As medições relativas ao processo são tratadas no tópico seguinte.

Processo de Teste

Apresenta considerações práticas sobre o processo de teste, como as medições do processo, reuso dos testes, geração de caso de teste.

2.2.5. Manutenção de software

Uma vez em operação anomalias são descobertas, alterações no ambiente operacional são feitas, e novos requisitos de usuários surgem. A fase de manutenção do ciclo de vida se inicia após a entrega, contudo as atividades de manutenção ocorrem bem antes (SWEBOK, 2004).

A área de conhecimento de manutenção no Swebok é dividida em 4 subáreas (Figura 8).

Fundamentos da manutenção de software

Introduz os conceitos e terminologia que formam o escopo da manutenção de software. Os tópicos provêm a definição e onde eles são empregados no processo de manutenção.

Pontos chave da manutenção de software

Apresenta alguns tópicos importantes relacionados à aspectos técnicos e gerenciamento da manutenção de software.

Processo de manutenção

Apresenta as referências e padrões utilizados para implementar o processo de manutenção.

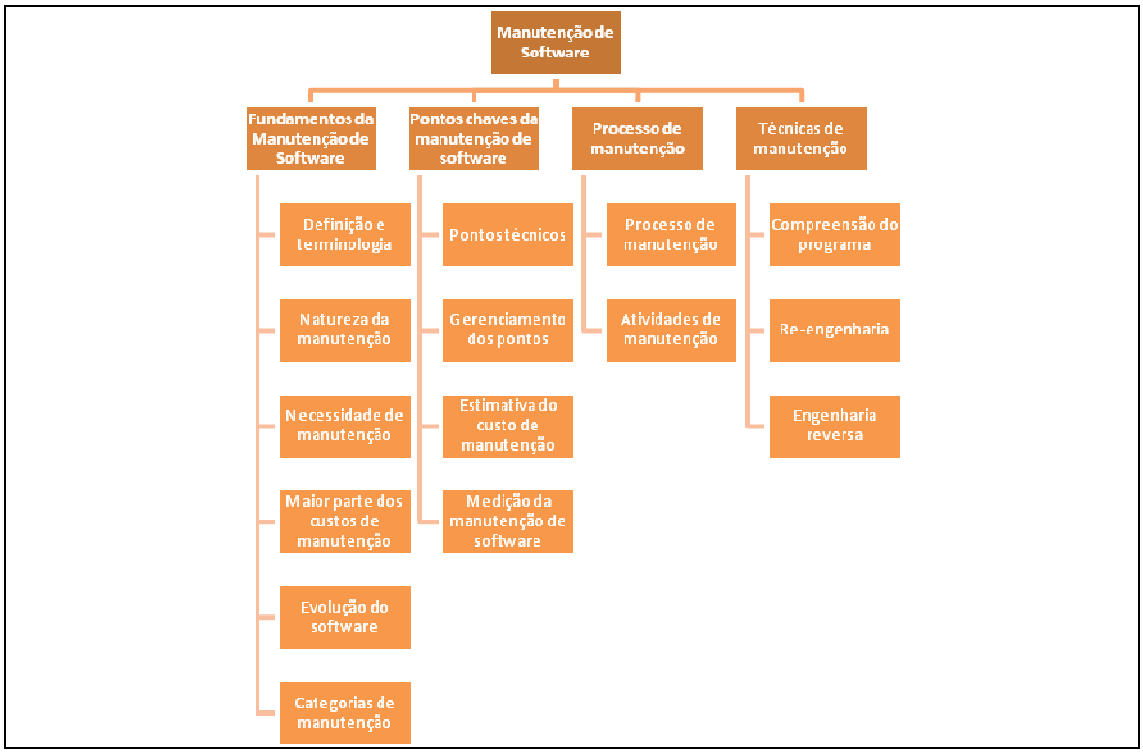

Figura 8 - Tópicos da área de conhecimento Manutenção de Software (SWEBOK 2004).

Técnicas de manutenção

Apresenta algumas técnicas aceitas no contexto da manutenção de software, como reegenharia, engenharia reversa, etc.

2.2.6. Gerência de configuração

A gerência de configuração é a disciplina que identifica a configuração do software em um ponto distinto no tempo, com o objetivo de sistematicamente controlar as mudanças, integridade e rastreabilidade (SWEBOK, 2004). Esta área de conhecimento está dividia em 6 seis sub áreas (Figura 9).

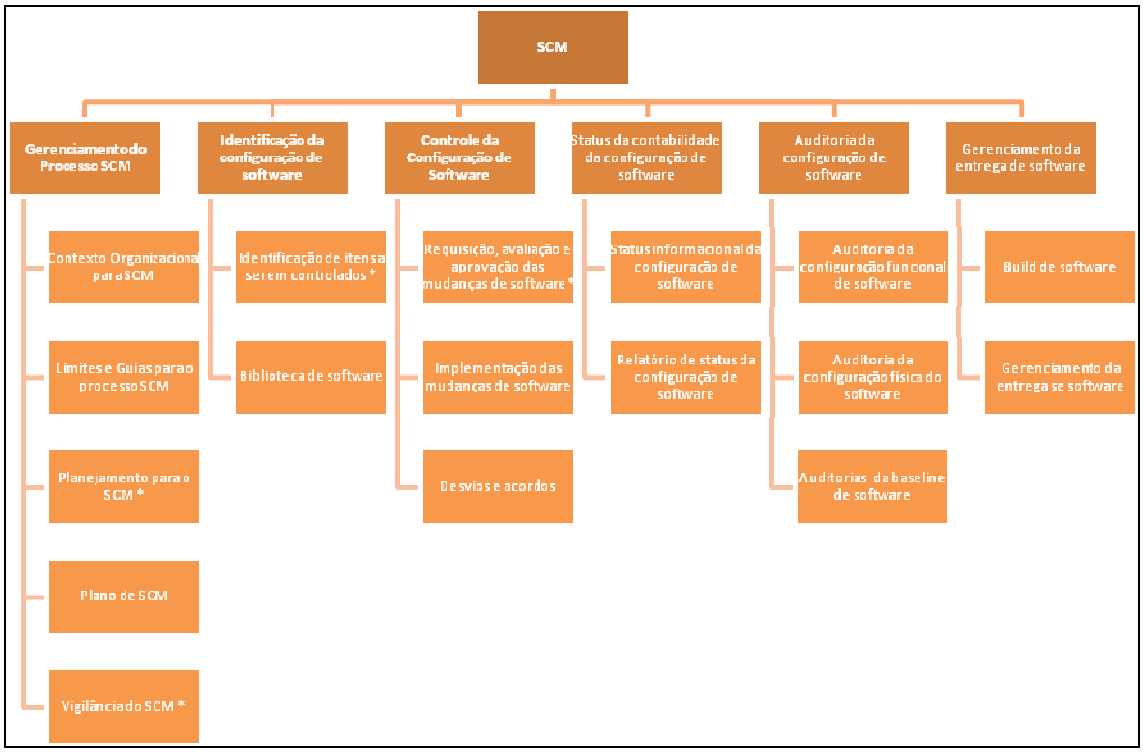

Figura 9 - Tópicos da área de conhecimento Gerência de configuração (SWEBOK 2004).

Gerenciamento do processo SCM

Cobre os tópicos do contexto organizacional, restrições e guias, planejamento, o plano e a vigilância do SCM.

Identificação da configuração de software

Esta subárea identifica os itens a serem controlados, o estabelecimento de esquemas para os itens e suas versões, estabelecimento de técnicas e ferramentas, utilizados para adquirir e controlar os itens de software.

Controle da configuração de software

Apresenta o gerenciamento das mudanças durante o ciclo de vida de software.

Status da contabilidade da configuração de software

Trata da gravação e relatórios da informação necessária para o efetivo gerenciamento da configuração de software.

Auditoria da configuração de software

Esta subárea trata das auditorias nos itens de configuração de software, podem ser física ou funcionais.

Gerenciamento da entrega de software

Trata da entrega da distribuição do software, tanto em distribuições internas quanto para os clientes.

2.2.7. Gerência da Engenharia de Software

Segundo o Swebok (2004, p. 119), a gerência de configuração trata da aplicação das atividades de gerenciamento – planejamento, coordenação, medição, monitoramento, controle e publicação – para garantir o desenvolvimento e manutenção do software de maneira sistemática, disciplinada e quantificada. No guia esta disciplina está organizada em 6 subáreas (Figura 10).

Inicialização de definição de escopo

O foco desta área é na determinação dos requisitos de software, através dos vários métodos de levantamento.

Planejamento do projeto de software

Esta subárea prevê o planejamento de processo, determinação de entregas, esforço cronograma e estimativa de custo, alocação de recurso, gerenciamento de risco, gerenciamento de qualidade e planejamento do gerenciamento.

Execução do Projeto de Software

Trata da implementação dos planos, gerenciamento dos contratos com fornecedores, implementação do processo de medição, monitoração de processos, controle e relatórios.

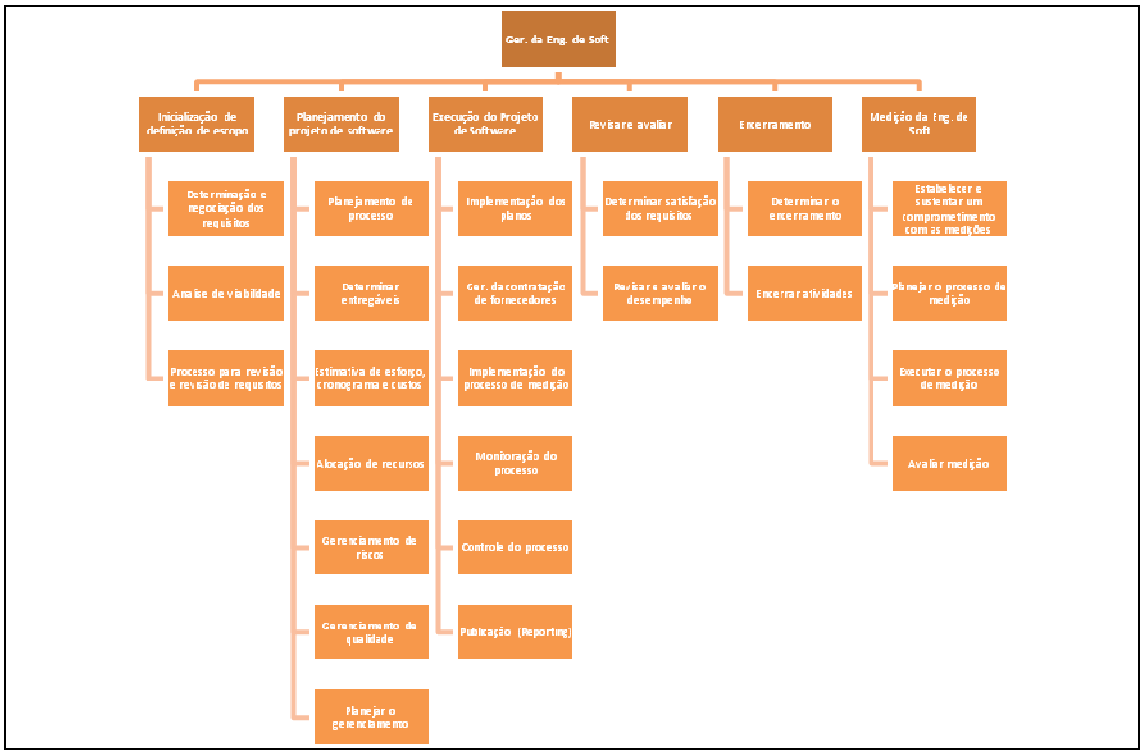

Figura 10 - Tópicos da área de conhecimento Gerência da Engenharia de Software (SWEBOK 2004).

Revisar e avaliar

Esta área trata da determinação da satisfação de requisitos, e revisão e avaliação do desempenho.

Encerramento

Apresenta as atividades para tratar o encerramento de projeto.

Medição da Engenharia de Software

Trata dos programas de medição para os produtos e processos de software.

2.2.8. Processo de engenharia de software

A área de conhecimento do processo de engenharia de software atua em dois níveis: A definição e implementação do ciclo de vida do software e na definição,

melhoria e avaliação de processos (SWEBOK 2004). Esta foi organizada em 4 subáreas (Figura 11).

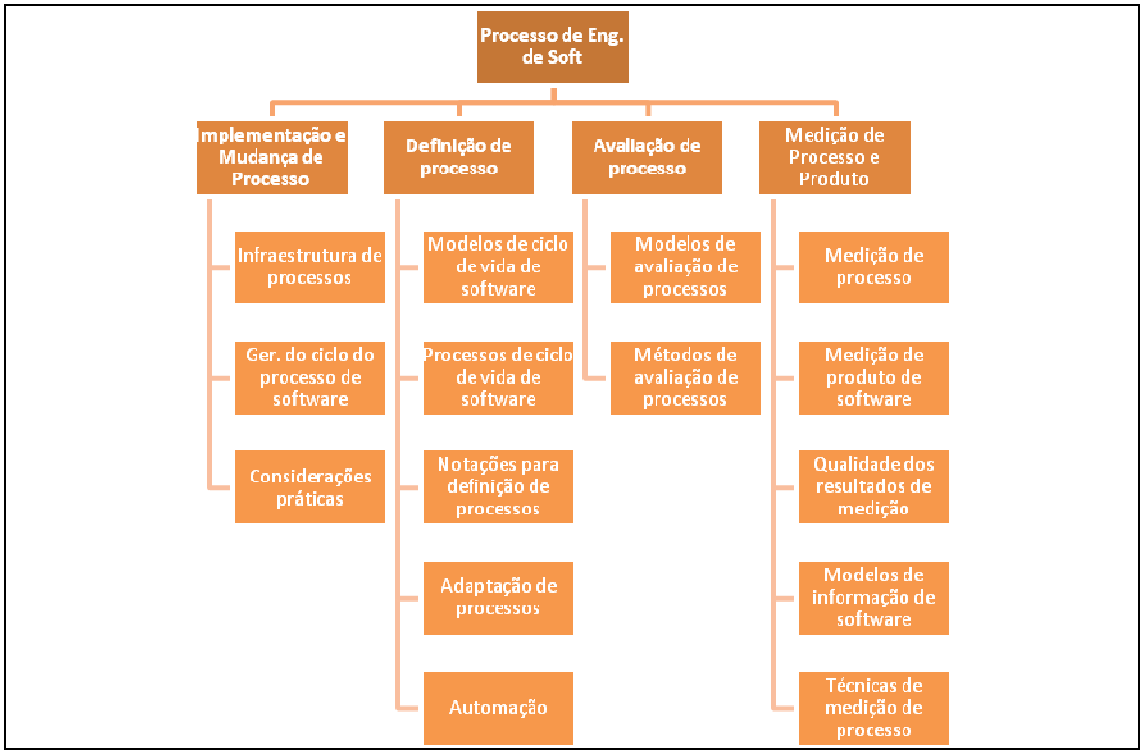

Figura 11 - Tópicos da área de conhecimento Processo de Engenharia de Software (SWEBOK 2004).

Implementação e Mudança de Processo

Esta subárea foca na mudança organizacional. Descreve a infraestrutura, atividades, modelos e considerações práticas de implementação do processo de mudança.

Definição de processo

A definição de processo pode ser um procedimento, uma política ou um padrão. Pode contemplar processos como: suporte, gerenciamento de processos, processo automatizado.

Avaliação de processo

Trata tanto do modelo como os métodos de avaliação de processos.

Medição de Processo e Produto

Trata dos itens relevantes da medição de processo e produto de software.

2.2.9. Métodos e Ferramentas de Engenharia de Software

Esta área de conhecimento trata de dois tópicos os métodos e as ferramentas. Os métodos trazem formalismo (estrutura) às atividades de engenharia de software com objetivo de executar estas atividades de forma sistemática e com sucesso. Já as ferramentas baseadas em computador com o objetivo de auxiliar o ciclo de vida de desenvolvimento de software (Figura 12).

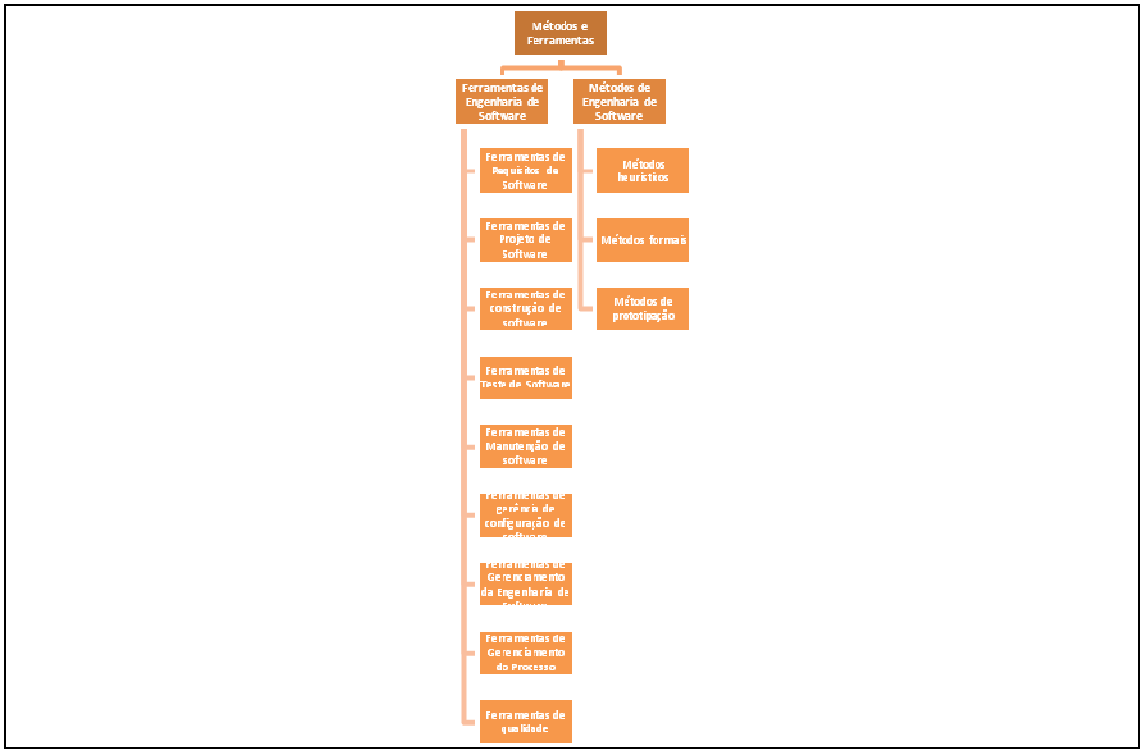

Figura 12 - Tópicos da área de conhecimento Métodos e Ferramentas de Engenharia de Software (SWEBOK 2004).

2.2.10. Qualidade de software

Procura endereçar os itens que são relacionados ao conceito de qualidade para o software, ou seja, atender os requisitos do software e alcançar a satisfação do cliente. Esta área cobre as técnicas estáticas, sem previsão de execução do software, e no guia está organizada em três subáreas (Figura 13).

Fundamentos da qualidade de software

Trata de conceitos fundamentais relacionados a qualidade como cultura e ética, custos, etc.

Processo de gerenciamento da qualidade de software

Define os processos, responsáveis, requisitos, medições e suas saídas e os canais de feedbak.

Considerações práticas

Trata de aspectos práticos da qualidade de software como requisitos e técnicas de gerenciamento da qualidade.

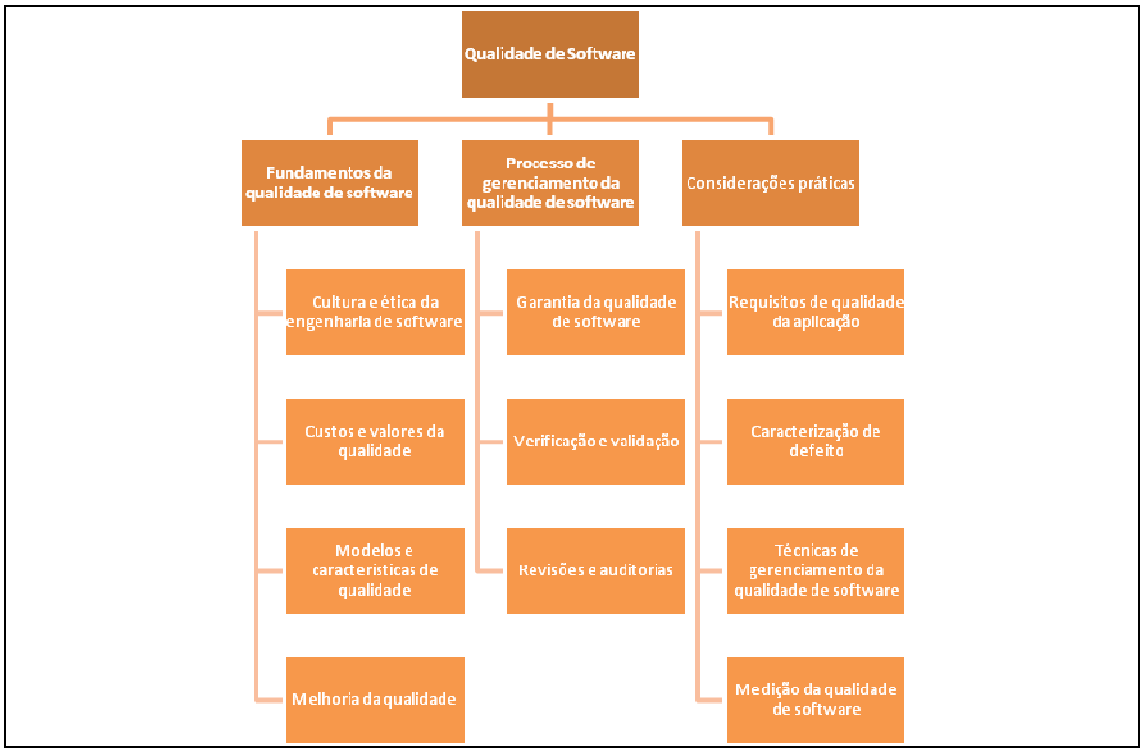

Figura 13 - Tópicos da área de conhecimento Qualidade de Software (SWEBOK 2004).

3. MÉTRICAS DE SOFTWARE

3.1. Introdução

As métricas de software são um importante recurso utilizado na engenharia de software para avaliar a qualidade e adequação de um produto de software, ou monitorar e controlar a execução de um processo.

Como já citado anteriormente no contexto da engenharia de software possuímos dois tipos métricas, as métricas de produto que têm por objetivo avaliar a adequação dos produtos de software a fatores de qualidade, como por exemplo, os fatores contidos na ISO/IEC-9126 - Qualidade Interna e Externa (Figura 14). Estas métricas podem ser aplicadas em itens como o código fonte, modelos (Estrutura dos componentes), o produto executável e a documentação.

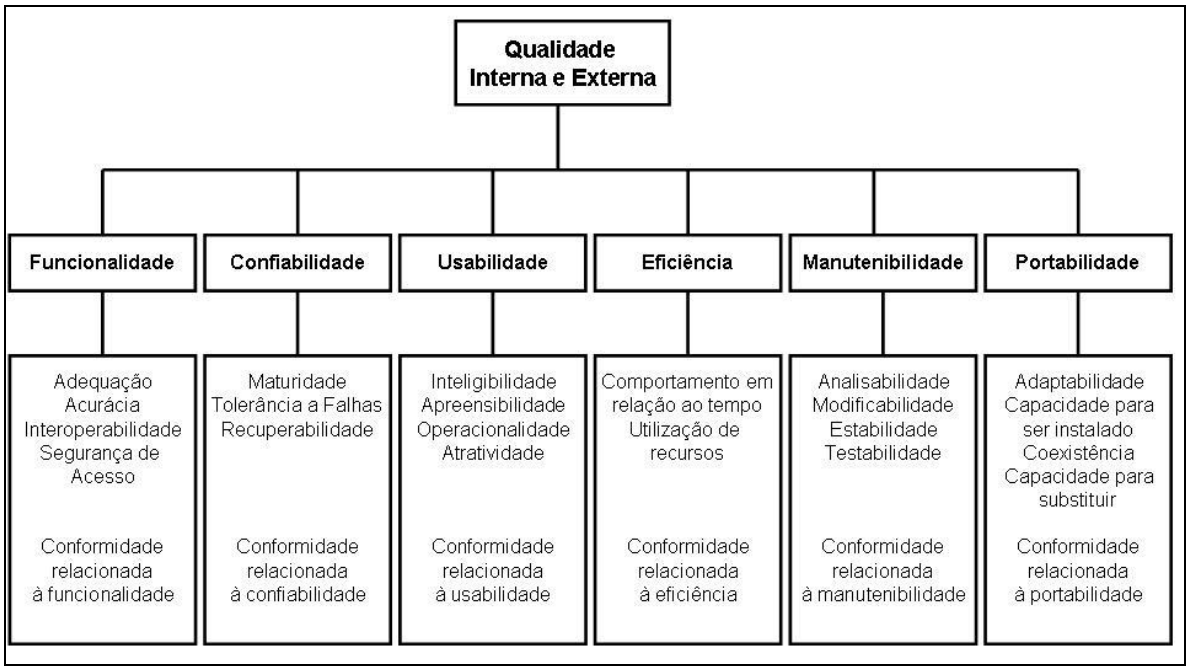

Figura 14 – Quesitos de Qualidade Interna e Externa da ISO/IEC-9126

Já as métricas do processo de software têm por objetivo acompanhar a execução de um processo com o objetivo de encontrar desvios. Usualmente fatores como produtividade, efetividade e acurácia são avaliados com este tipo de métrica.

3.2. Conceitos

Quando estamos tratando de métricas de software é importante ter bem identificado alguns conceitos sobre medições que podem gerar confusões de entendimento.

Ao falarmos em medir um produto ou processo de software podemos estar tratando de três itens: As medidas, as métricas e os indicadores. Segundo Pressman (2006) estes três termos são categorizados da seguinte forma:

- **Medida**: Fornece uma indicação quantitativa da extensão, quantidade, dimensão capacidade ou tamanho de algum atributo de um produto ou processo. Ex.: Número de defeitos e uma função do sistema;
- **Métrica**: Uma medida quantitativa do grau em que um sistema, componente ou processo possui um determinado atributo. Ex.: Número médio de erros encontrados por revisão, Número médio de erros encontrados no teste de unidade;
- **Indicador**: É uma métrica ou combinação de métricas que fornece profundidade na visão do processo de software, projeto de software ou produto em si. Ex.: Número médio esperado de detecção de defeitos nos testes unitários.

As métricas podem utilizar alguns recursos matemáticos para transformar em números (com significado) informações colhidas dos produtos ou processos de software. Com certa frequência identificamos a razão, a proporção, o percentual e a taxa como as técnicas matemáticas mais empregadas. Temos como exemplo de aplicação destas técnicas:

- **Razão**: (N° de desenvolvedores/ N° de Testers) X 100%
- **Proporção**: N° de clientes satisfeitos/ N° total de clientes de um produto de software

- **Percentual**: % de defeitos relacionados a uma etapa do desenvolvimento
- **Taxa**: Taxa de Defeito= (N° de defeitos/Oportunidades de Falhas) x K*
- 1000 linhas de código

3.3. O método GQM

Apesar do formalismo aplicado às métricas e a necessidade da utilização de recursos matemáticos, esta não é a parte mais complicada do processo. Antes de se pensar nas métricas são necessários objetivos e metas claros que precisam ser endereçados por estas medições.

Pensando nisso foi que Basili, Caldiera e Rombach desenvolveram um método chamado GQM (Goal-Question-Metric), que organiza o paradigma de medição de software em três níveis (GQM, 2009):

- **Nível conceitual**: Definição de metas de medição envolvendo produtos, processos e/ou recursos (Goal).
- **Nível operacional**: Definição de questões que tentem caracterizar o objetivo da medição no contexto de um problema de qualidade de um ponto vista específico (Question).
- **Nível quantitativo**: Associar a cada questão um conjunto de dados, tanto subjetivos como objetivos, que ajudam a prover uma resposta quantitativa à questão (Metric).

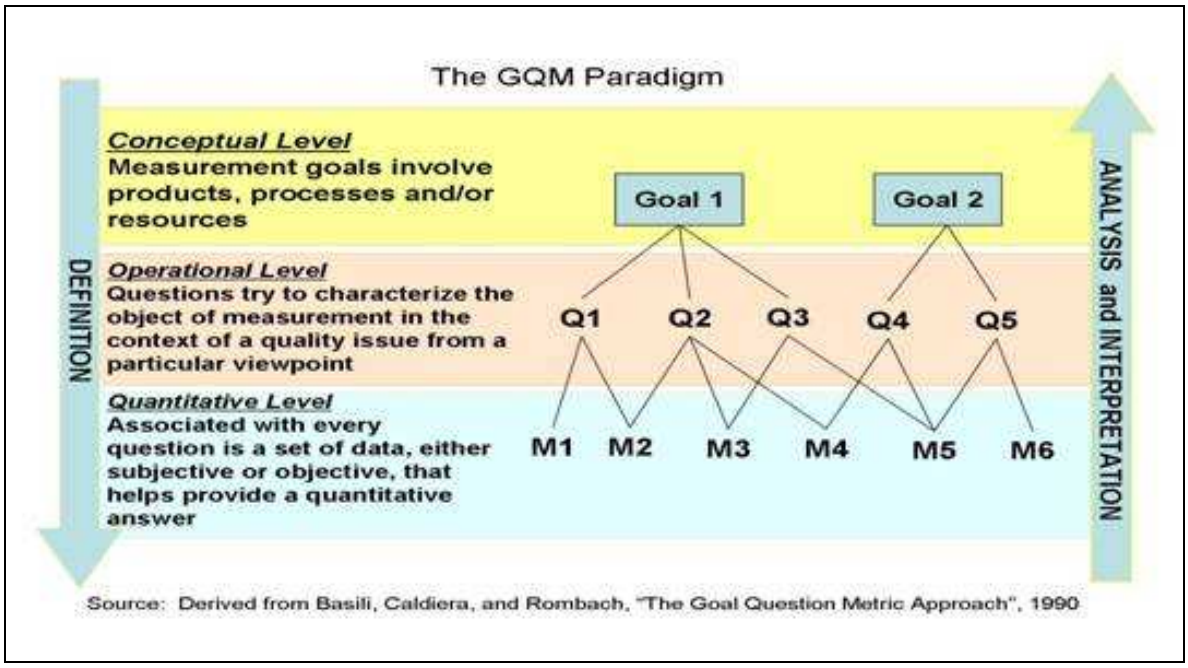

Figura 15 – O paradigma GQM (GQM, 2009).

Segundo os autores, a utilização do método GQM pode ser melhor compreendida se tratada em 5 passos fundamentais, que são (GQM, 2009):

1. Desenvolver um conjunto de metas de negocio, corporativas e de projetos, para produtividade e qualidade;
2. Gerar questões (baseada em modelos) que definam estas metas da maneira mais completa possível e quantificável;
3. Especificar as necessidades de medições a serem coletadas para responder estas questões e rastrear a conformidade do produto e processo com as metas;
4. Desenvolver mecanismos para a coleta de dados;
5. Coletar, validar e analisar os dados em tempo real provendo um feedback para os projetos para a tomada de ações corretivas;
6. Analisar os dados de uma maneira *postmortem* para avaliar a conformidade com as metas e fazer recomendações de melhorias futuras.

3.4. As normas ISO

Dentro do contexto de medição de software existem algumas normas da ISO (International Organization for Standardization) que tratam do assunto tanto nos seus aspectos técnicos como de gerenciamento e processuais. Como já citado anteriormente a norma ISO/IEC-9126 (Qualidade do Produto de Software) trata das características e subcaracterísticas de qualidade bem como métricas que podem ser utilizadas. Esta norma trata de três tipos de avaliação:

- **Qualidade Interna**: Aplicada na etapa de desenvolvimento e nos os itens não executáveis;
- **Qualidade Externa**: Aplicada na etapa de testes e nos itens executáveis do software;
- **Qualidade em Uso**: Procura avaliar o quanto o produto atende aos requisitos do usuário em seu ambiente previsto de uso.

Ainda segundo esta norma as medições estão interrelacionadas com a qualidade geral dos produtos e processo de desenvolvimento de software, conforme detalhado na Figura 16.

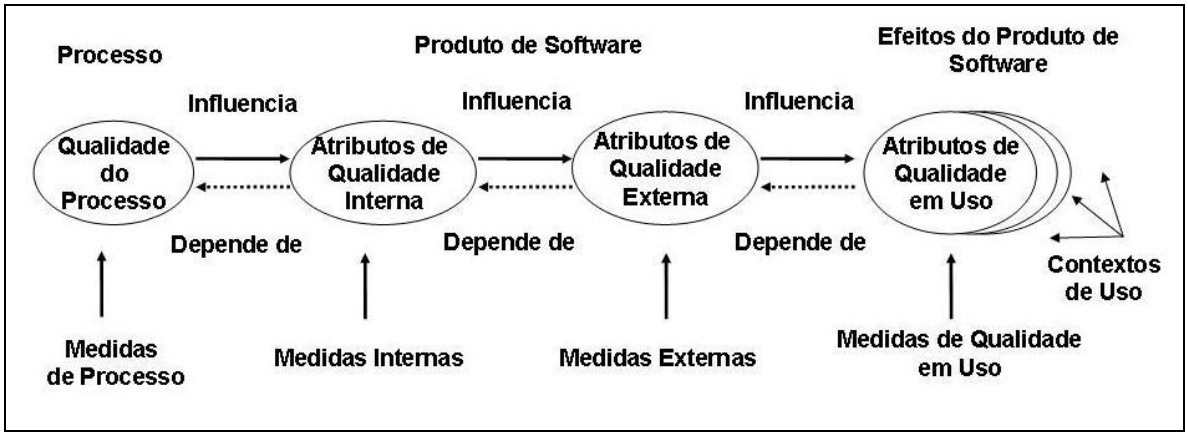

Figura 16 – ISO/IEC-9126 Relacionamento entre os tipos de medidas (ABNT, 2003).

Já a série de normas ISO/IEC 14598 (Avaliação dos Produtos de Software) trata dos procedimentos e métodos do processo de medição e avaliação da qualidade do produto de software.

Contudo como o assunto de medições estava disperso e tratado de forma independente em duas normas, surgiu a necessidade de criação de uma nova norma que agrupasse todos estes conceitos de maneira integrada e concisa. Daí surgiu a ISO/IEC 25000, também conhecida como SQuaRE (Software product Quality Requirements and Evaluation). Esta norma está organizada em 5 grandes módulos:

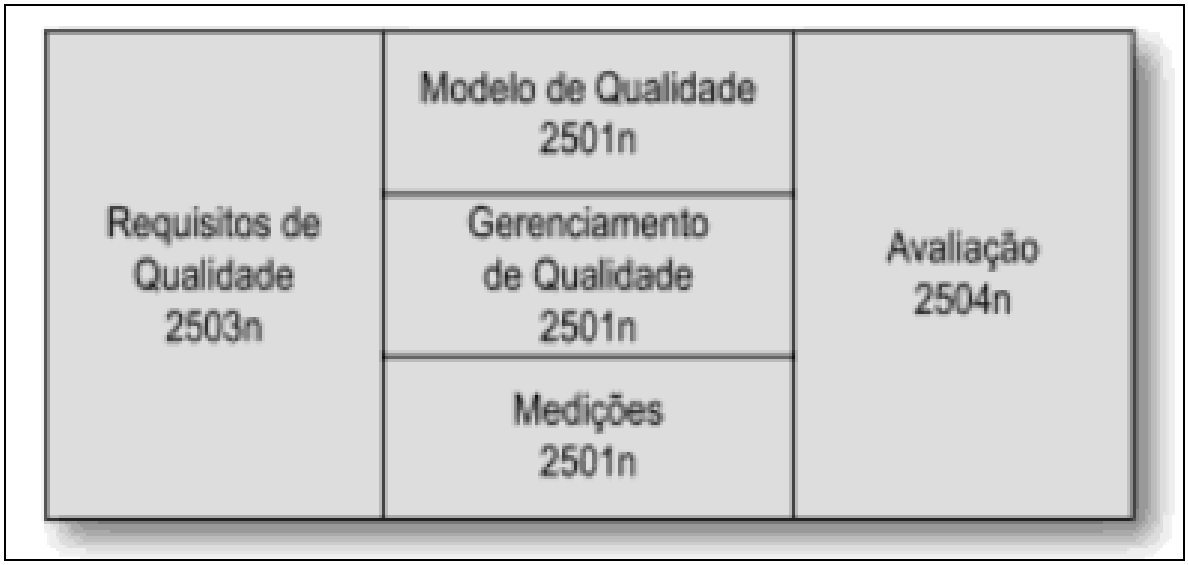

Figura 17 – Organização da norma ISO/IEC 25000.

- **Gerenciamento de qualidade**: Trata da definição de termos gerais e recomendações e sugestões de como utilizar o modelo;
- **Modelo de qualidade**: Traz boa parte da ISO/IEC 9126-1, conceitos de qualidade interna, externa e em uso;
- **Medição**: Defini e descrever os aspectos relacionados a esta atividade, com apresentação de métricas;
- **Requisitos de qualidade**: Estabelece os objetivos de qualidade;
- **Avaliação**: Prevê a avaliação e confronto dos resultados com o modelo definido.

3.5. Métricas de Produto de Software

Nesta seção serão apresentadas algumas métricas relacionadas às medições do produto de software. Estas métricas em sua grande parte foram retiradas da norma ISO/IEC 9126 (2 e 3) e estão relacionadas a uma das 6 características de qualidade prevista nesta norma.

As métricas vão ser apresentadas semelhantes ao modelo GQM para os níveis operacionais e quantitativo, questões relacionadas às características de qualidade, com métricas para endereçar cada uma destas questões.

3.5.1. Funcionalidade

A métrica a seguir trata da verificação da adequação da implementação do software em relação às funções previstas na especificação de requisitos.

Quesito	Descrição
Pergunta	Quão completa está a implementação em relação à especificação de requisitos?
Método	Executar os testes funcionais, com base na ER, e contar o número de funções que não estão implementadas
Formula	X = 1 – A/B A – Número de funções faltantes B – Número de funções da ER
Análise do resultado	0<=X<=1 (Quanto mais próx. de 1 melhor)

Tabela 1 – Métrica de avaliação da adequação da implementação dos requisitos (ABNT, 2003).

Outra métrica que poderia ser aplicada para a categoria de funcionalidade, seria a que verifica quão correta está a implementação das interfaces do sistema, relacionando-se assim a subcaracterística de interoperabilidade.

Quesito	Descrição
Pergunta	Quão correta está a implementação dos protocolos de interface?
Método	Contar o número de interfaces que foram implementadas conforme as especificações
Formula	X = A/B A – Número de interfaces implementadas de forma consistente B – Número de interfaces a serem implementadas
Análise do resultado	0<=X<=1 (Quanto mais próx. de 1 melhor)

Tabela 2 – Métrica de avaliação da implementação das interfaces de um sistema (ABNT, 2003).

3.5.2. Confiabilidade

Uma das métricas mais comuns relacionada ao quesito confiabilidade é uma das que trata da maturidade do software, o MTBF (Mean Time Between Failure) ou Taxa Média Entre Falhas. Esta métrica procura avaliar com que freqüência o software falah quando está em operação.

Quesito	Descrição
Pergunta	Com que frequência o software falha em operação?
Método	Contabilizar as falhas durante um período de operação e computar a média dos intervalos entre as falhas
Formula	X = T1/A Y = T2/A A – Número total de defeitos identificados (no período observado) T1 – Tempo de operação

	T2 – Soma do intervalo de tempo entre a ocorrência de falhas consecutivas
Análise do resultado	0<X,Y (Quanto maior melhor, maior tempo entre falhas)

Tabela 3 – Métrica MTBF para software (ABNT, 2003).

3.5.3. Usabilidade

A usabilidade também é um quesito que poder ser medido e avaliado para melhoria do produto de software. Um exemplo é a avaliação da operação do software, com a verificação da implementação de validação de dados de entrado no produto.

Quesito	Descrição
Pergunta	Qual a proporção de itens de entrada que possuem validação de dados?
Método	Contar o número de itens de entrada em que é possível validar os dados, e contar os itens em que não é possível validar a entrada
Formula	X = A/B A – Número de itens de entrada que são validados B – Número de itens de entrada que não são validados
Análise do resultado	0<=X<=1(Quanto mais próximo de 1 melhor)

Tabela 4 – Métrica de avaliação da validação de dados de entrada (ABNT, 2003).

3.5.4. Eficiência

Um das formas de se analisar a eficiência de um software ou produto de software é através da avaliação do Throughput. Esta métrica permite identificar quantas tarefas foram executadas em um período de tempo de execução do software.

Esta métrica é muito utilizada em testes de carga e desempenho com o objetivo de validar se os requisitos não funcionais do software estão sendo cumpridos.

Quesito	Descrição
Pergunta	Quantas tarefas podem ser executadas com sucesso em um período de tempo?
Método	Iniciar vários Jobs de tarefas e verificar o tempo que leva para completar a operação
Formula	X = A/T A – Número de tarefas completadas com sucesso T – Período de tempo observado
Análise do resultado	0<X (Quanto maior melhor)

Tabela 5 – Métrica Throughput para avaliação da eficiência do software (ABNT, 2003).

3.5.5. Manutenibilidade

Até aspectos relacionados à manutenção de software podem ser avaliados. Um dos pontos mais interessantes é saber qual o impacto de uma mudança no software. Dependendo do resultado esta métrica pode gerar várias atividades de melhoria de processo ou até mesmo uma reavaliação da arquitetura, documentação e códigos fontes do software.

Quesito	Descrição
Pergunta	Com que frequência ocorrem impactos adversos após uma modificação?
Método	Contabilizar o número de impactos adversos depois das modificações e compará-lo com o número total de modificações

Formula	X = 1-A/B A – Número de impactos adversos detectados após modificações B – Número de modificações feitas
Análise do resultado	0<=X<=1 (Quanto mais perto de 1 melhor)

Tabela 6 – Métrica de avaliação de impacto das manutenções (ABNT, 2003).

3.5.6. Portabilidade

Até quesitos relacionados a portabilidade de um software podem ser avaliados, como por exemplo, avaliado o quão fácil é adaptar um software a um novo ambiente ou plataforma.

Quesito	Descrição
Pergunta	Pode o usuário ou o mantenedor facilmente adaptar o software a um ambiente?
Método	Observar o comportamento do usuário ou mantenedor em uma tentativa de adaptar o software ao ambiente
Formula	T = Soma do tempo gasto para completar a adaptação
Análise do resultado	0< T (Quanto menor melhor)

Tabela 7 – Métrica de avaliação da capacidade de portabilidade do software (ABNT, 2003).

3.5.7. Outras Medições

Além das métricas relacionadas aos seis quesitos de qualidade previstos na norma ISO/IEC 9126, outras medições também são importantes no processo de software, pois podem trazer informações importantes principalmente sobre o tamanho de complexidade do produto de software.

Um dos conjuntos de medições e métricas para estes objetivos, com foco na orientação objeto, foi o desenvolvido por Chidamber e Kemerer, apelidadas de conjunto de métricas CK. Estas métricas são baseadas em classes, e trazem várias informações sobre tamanha, acoplamento e coesão das classes de sistema OO (KAN, 2002):

- Métodos ponderados por classes (WMC – *Weighted methods per class*): o VMC é a soma das complexidades dos métodos de uma classe, onde a complexidade é medida por complexidade ciclomática;
- Profundidade da árvore de herança (DIR – *Depth of the inheritance tree*): É o tamanho máximo do caminho de hierarquia de uma classe do nodo até a raiz de uma árvore de herança;
- Número de filhos (NOC – *Number of children*): Número de sucessores imediatos de uma classe na hierarquia (Subclasses);
- Acoplamento entre as classes de objetos (CBO – *Coupling between object classes*): Número de classes em que uma determinada classe está acoplada;
- Resposta de uma classe (RFC – *Response for a class*): Número de métodos que podem ser executados em resposta a uma mensagem recebida por objeto ou classe;
- Falta de coesão em métodos (LCOM – *Lack of cohesion in methods*): A coesão de uma classe é indicada por quão próximo os métodos locais estão relacionados a instâncias locais de variáveis em uma classe.

3.6. Métricas do Processo de Software

O processo de desenvolvimento de software, como outros processos de engenharia, podem e devem ser medidos com o objetivo de controlar e monitorar sua execução a procura de desvios ou oportunidades de melhoria. Esta monitoração e acompanhamento pode trazer benefícios como:

- Obter informações sobre custos
- Retorno do investimento
- Identificar pontos que precisam de melhoria (SPI)

- Avaliar o efeito ou resultado de uma melhoria (SPI)
- Avaliar a produtividade
- Identificar o comportamento e localização dos defeitos

Há algumas décadas um conjunto de técnicas propostas por Ishikawa se tornaram muito comuns no âmbito da engenharia e processos industriais e também na área de desenvolvimento de software. Estas técnicas denominadas controle estatístico de processo, são utilizadas para identificar e remover variações no processo que excedam as variações esperadas de causas naturais. O propósito do controle de processo é detectar qualquer anomalia no processo.

Estas técnicas e/ou ferramentas seriam num total de 7: as planilhas de verificação, diagramas de Pareto, histogramas, gráficos de execução, digramas de dispersão, gráficos de controle e diagrama de causa e efeito (Espinha de peixe) (Figura 18).

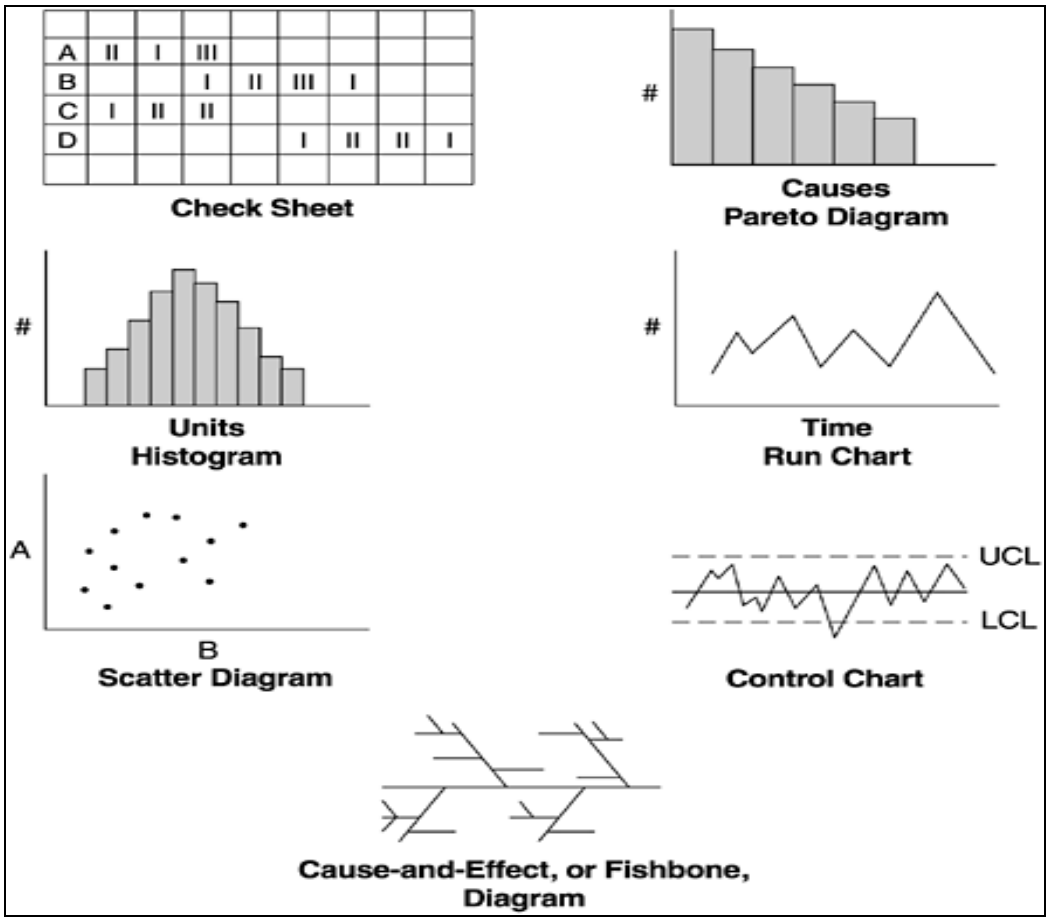

Figura 18 - Sete ferramentas básicas de controle da Qualidade de Ishikawa (Kan, 2002).

A seguir serão apresentadas algumas métricas usualmente aplicadas ao processo de desenvolvimento de software. Estas métricas também seguem o padrão do GQM.

3.6.1. Projeto

Uma das métricas mais aplicadas a projetos de desenvolvimento de software são as que tratam da taxa de acerto em relação a cronograma e esforço.

Quesito	Descrição
Pergunta	Qual é a taxa de acerto do cronograma/esforço do projeto?
Método	Contabilizar a o cronograma/esforço e comparar com o planejado

Formula	ACP = Duração Atual do Projeto / Duração Estimada do Projeto ACE = Esforço Atual do Projeto / Esforço Estimado do Projeto
Análise do resultado	Quanto mais próximo de 1 melhor

Tabela 8 – Métrica de avaliação da taxa de acerto de cronograma e/ou esforço do projeto.

3.6.2. Testes

O processo de teste de software é um dos que permite a aplicação de vários tipos de métricas. Notadamente temos um destaque para as métricas de efetividade que procuram avaliar a capacidade dos testes de detectar defeitos.

Uma das métricas mais utilizadas é a de efetividade de remoção de defeitos, que avalia a quantidade de defeitos removidos em uma determinada etapa de teste em comparação com as etapas seguintes. Esta métrica deve ser utilizada com cuidado, pois ela pode apresentar desvios se aplicada com o projeto ainda em andamento.

Quesito	**Descrição**
Pergunta	Quão efetivo é determinada fase dos meus testes?
Método	Executar os testes e contabilizar a quantidade de defeitos detectados em cada fase
Formula	ERD = Defeitos Removidos Durante uma Fase / (Defeitos Removidos Durante uma Fase + Defeitos Encontrados Depois) * 100%
Análise do resultado	Quanto mais próximo de 100% melhor

Tabela 9 – Métrica de avaliação da efetividade de uma fase de teste.

Uma outra métrica que também avalia a efetividade do processo de teste é que a avalia a efetividade da detecção antes da entrega para o cliente.

Quesito	Descrição
Pergunta	Como está a efetividade de detecção dos defeitos antes da entrega?
Método	Contabilizar o número de defeitos antes e após a entrega
Formula	TDCE = Número de defeitos antes da entrega / (Número de defeitos antes da entrega + Número de defeitos após a entrega)
Análise do resultado	Quanto mais próximo de 1 melhor

Tabela 10 – Métrica de avaliação da efetividade de detecção de defeitos antes da entrega para o cliente (KAN, 2002).

3.6.3. Manutenção

Já para o processo de manutenção uma das métricas mais comuns é a avaliação do backlog, esta métrica avalia se o ritmo de atendimento às demandas de manutenção está adequado em relação à quantidade de demandas que são recebidas.

Quesito	Descrição
Pergunta	Como está o atendimento às demandas de correção?
Método	Contabilizar o número de demandas atendidas e as que chegaram no mês
Formula	IGB = Número de Problemas Encerrados / Número de Problemas que Chegaram * 100% (Por mês)

Análise do resultado	Quanto mais próximo de 100% melhor

Tabela 11 – Métrica de avaliação de backlog de demandas de manutenção.

Esta é uma métrica que pode, por exemplo, ser exibida em um gráfico de controle com limites superiores e inferiores, para garantir que o esforço está adequado ao volume de demandas que são recebidas (Figura 19).

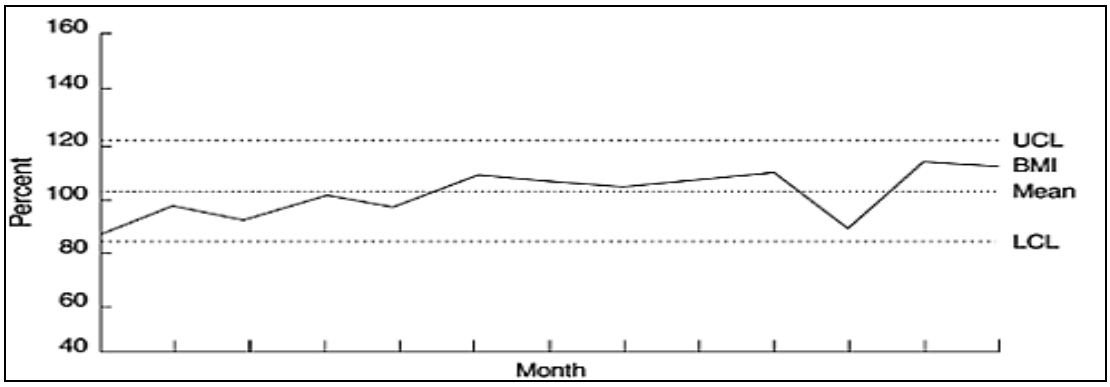

Figura 19 – Exemplo de gráfico de Backlog (KAN, 2002).

REFERÊNCIAS

1. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 9126-1: Engenharia de Software – Qualidade do Produto. Rio de Janeiro: ABNT, 2003, 21 p.

2. IEEE Software Engeneering Coordinating Commitee. SWEBOK: Guide to the Engeneering Body of Knowledge – Version 2004. Disponível em: < http://www.swebok.org.>

3. PRESSMAN, Roger S. Engenharia de Software. 6. ed. McGraw-Hill, 2006.

4. SOMMERVILLE, IAN. Engenharia de Software. 8. ed. Addison Wesley, 2007

5. KAN, Stephen h. Metrics and Models in Software Quality Engineering. Second Edition. Addison Wesley, 2002.

6. MUNSON, Software Engineering Measurement. Auerbach Publications, 2003.

7. KOSCIANSKI, André; SOARES, Michel dos Santos. Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software, 2006.

8. GQM *in* Gold Practices. - https://www.goldpractices.com/practices/gqm/