UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EDNA SATIKO EIRI TREBIEN

PROPOSTA DE METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE VOLTADAS À EDUCAÇÃO

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

VITÓRIO BRUNO MAZZOLA, DR.

Florianópolis, maio de 2002

PROPOSTA DE METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE VOLTADAS À EDUCAÇÃO

EDNA SATIKO EIRI TREBIEN

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração em Sistemas de Computação, na linha de pesquisa em Engenharia de Sistemas Distribuídos e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

FERNANDO OSTUNI GAUTHIER, Dr.
Coordenador do Curso

Banca Examinadora

VITÓRIO BRUNO MAZZOLA, Dr.
Presidente

ROBERTO WILLRICH, Dr.
Membro

MURILO SILVA CAMARGO, Dr.
Membro

A síndrome do 99% certo, significa:

- 3 palavras datilografadas com erros/página;
- sem telefone ou luz durante 15 minutos/dia;
- 17.000 bebês recém-nascidos deixados cair por médicos ou enfermeiras/ano;
- 100.000 pessoas tomando medicação errada/ano;
- beber água não potável durante 3 dias/ano;
- 1,4 milhões de pessoas morrendo por alimentos envenenados/ano;
- 2 aterrissagens e/ou decolagens inseguras por dia em Cumbica;
- 500 operações cirúrgicas incorretas/semana;
- 10.000 cheques descontados na conta errada/dia;

(FIESC, 1994)

A meus filhos, Guto e Tonia.

Agradeço imensamente à profª Fahena P. Horbatiuk, pelo incentivo, amizade e cooperação. Ao prof. Jefferson pelo carinho e companheirismo, a Cristina Machado pelo coleguismo e colaboração, ao Antonio Carlos pelo incentivo e as palavras amigas, ao professor Dr. Vitório Bruno Mazzola, pela orientação e porto seguro nas horas difíceis, a Erna Trebien pela "mãezona" que sempre será, aos colegas e a todos que colaboraram para a realização deste trabalho.

SUMÁRIO

RESUMO

O cenário atual da indústria de software, sedimentada desde 1993 até a última pesquisa de 1999, é apresentado para parametrizar, em dados estatísticos, a evolução desse mercado, fundamentados no relatório bienal da pesquisa realizada pelo Ministério da Ciência e Tecnologia-MCT, sob a responsabilidade da Secretaria de Política de Informática e Automação-SEPIN, no âmbito do Subcomitê Setorial da Qualidade e Produtividade em Software, do Programa Brasileiro da Qualidade e Produtividade, a qual retrata numericamente os resultados da avaliação da gestão da qualidade nas empresas, a sua evolução e os procedimentos específicos para a qualidade de software. Dentre os inúmeros esforços e trabalhos desenvolvidos para a melhoria de processos de softwares, são apresentados e estudados, com mais profundidade, aqueles que, no momento atual, são objeto de estudo e padronização pela ISO/IEC e pela ABNT, como ferramentas de avaliação de gerenciamento e Engenharia de Software. São apresentadas as características das normas e modelos, contribuindo para a análise e seleção de instrumentos de melhoria e subsídios para elaboração de planos de qualificação de processos de desenvolvimento de softwares, compondo um conjunto de ferramentas de avaliação do produto e processo. Dessa forma, a proposta da metodologia de processo de software educacional apresenta-se como um segmento primordial na aplicação, teste e validação de softwares qualitativos, com interfaces amigáveis, tanto para o educador como para o educando. Atualmente, a proliferação de softwares educacionais, quantitativamente, torna-se um atraente mercado de estudos de validação e avaliação e, principalmente, adequação, como instrumento de ensino-aprendizagem.

ABSTRACT

The present scenario of the software industry sedimentary from 1993 to the last survey in 1999, is presented to transform the evolution of this market into statistical data grounded in the biannual report of the research, made by the MCT-Ministério da Ciência e Tecnologia (Ministry of Science and Technology), under the responsibility of SEPIN-Secretaria de Política de Informática e Automação (Department of Computing Policy and Automation Office), in the scope of the Subcomitê Setorial de Qualidade e Produtividade (Quality and Productivity Subcommittee) in software of the Brazilian Quality and Productivity Program, which shows in numbers the results of the evaluation of the quality management in companies, its evolution and specific procedures for the software quality. Among the efforts and papers developed for the improvement of software processes, those which at present are the subject of study and standardization by ISO/IEC and ABNT as evaluation tools for software management and engineering, are presented and studies more deeply. The characteristics of the norms and models are presented, contributing to the analysis and selection of improvement and subsidy instruments to make plans to qualify the software development process **forming a set of evaluation tolls for product and process**. Thus, the proposal of the educational software methodology is presented as a primordial segment in the application, testing and validation of qualitative software with friendly interfaces for the educator and the student. Nowadays, the proliferation of educational software makes it an attractive market for validation studies and evaluation and specially its adequacy as a teaching-learning instrument.

1. INTRODUÇÃO

À medida que se presencia, cada vez mais, a utilização de produtos de softwares em processos de ensino-aprendizagem, aprimoram-se os instrumentos de análise e subsídios para sua validação, levando a uma conscientização maior quanto à garantia da qualidade, e qualificação desses instrumentos, como equipamentos de melhoria e suporte.

Levando em conta que os ambientes de desenvolvimento de softwares, em sua maioria, são abordagens quase artesanais, em que a habilidade individual prevalece sobre as ferramentas e técnicas, o avanço dos esforços para mudar esse panorama é notado pelos resultados apresentados neste trabalho, em que se pretende avaliar e mostrar um ambiente mais racional de desenvolvimento de processos e produtos de software, voltados ao processo de ensino-aprendizagem.

Dessa forma, pretende-se desenvolver uma proposta metodológica que privilegie o embasamento em padrões internacionais de engenharia de software, contribuindo para uma nova forma de organização e sofisticação dos processos de ensino-aprendizagem, atendendo às novas demandas educacionais, baseadas em educação permanente e aprendizagem cooperativa. Não se pretende aqui discutir ou fomentar uma discussão das melhores filosofias de ensino-aprendizagem e nem a sociedade como entidade interdependente. O escopo da pesquisa alcança a definição de processo de software padronizado e efetivo, para desenvolvimento de software educacional baseado na tríade: Pessoas-Processo-Tecnologia.

A introdução e seleção de software educativo se dá através daqueles que “estão disponíveis”, ou “são compatíveis com as equipes de trabalho”, ou “têm a ver com o que se está a ser ensinado”, e não através da definição de critérios específicos para avaliar a qualidade. Nesse contexto, é essencial facilitar o acesso ao conhecimento de critérios que possam fornecer subsídios para um processo de desenvolvimento ou avaliação de software, com um nível de segurança adequado, de que o item ou produto esteja dentro de parâmetros técnicos confiáveis (GAMEZ, 1998).

Por outro lado, um software educacional possui características particulares e sua utilidade é avaliada por meio de sua didática e pedagogia. Ele deve estar inserido em um contexto de aprendizado pré-definido e proporcionar autonomia, cooperação,

criatividade, pensamento crítico, descoberta e construção do conhecimento. Dessa forma, além de ser intuitivo, fácil e eficiente de usar, o software educacional deve ser didático (CYBIS, 1997). Além disso, deve envolver a relação professor-aluno-ensino-aprendizagem, possibilitando a exploração da informação num ambiente contextualizado e interativo. Um software voltado para a educação deve refletir esse contexto em sua construção, com a garantia de sua qualidade. A presente proposição metodológica orienta um processo de desenvolvimento e construção, baseado em padrões internacionais de qualidade.

No capítulo dois, busca-se modelar numericamente o retrato da realidade atual, utilizando-se dos relatórios do MCT/SEPIN sobre Qualidade no setor de software brasileiro, que vem sendo realizado desde 1993, com periodicidade bienal, sendo o último em 1999, com dados ainda a serem publicados. Traça-se um paralelo entre a indústria nacional e internacional de software, identificando as semelhanças e as atitudes tomadas para a solução dos problemas que envolvem o desenvolvimento de software e a relação com os usuários.

No capítulo três, são tratados os métodos e técnicas de desenvolvimento de software, com um breve histórico de sua evolução, principalmente a descrição das normas internacionais em que se baseia a proposta. Todos os modelos e normas, de produto ou de processo, são subsídios para diminuir a influência, no resultado final, das habilidades individuais e dinamizar o nível de habilidade sedimentada em técnicas, automatização de processos e tecnologia de desenvolvimento e gerenciamento de produtos de softwares.

Este trabalho também está fundamentado em pesquisa documental eletrônica via páginas da Internet, em que se procurou determinar o panorama atual da conscientização e procura de soluções para resolver a crise do software, vivida constantemente, principalmente no Brasil. Com a participação em fóruns de debates e conferências nacionais e internacionais sobre o assunto, procurou-se determinar como está sendo solucionada essa crise pelas universidades, empresas e centros de pesquisa do país. Pela pesquisa bibliográfica, procurou-se avaliar este mercado, ainda em seus passos iniciais, bastante distante da qualidade da indústria, por exemplo; demonstrar o nível de conscientização mundial sobre o tema e os esforços dos países-pólos desenvolvedores de softwares globais. Pelas análises feitas em mais de vinte anos de

vivência pessoal no desenvolvimento de softwares para *mainframes*, microcomputadores e interfaces, atuando em um mercado competitivo e altamente corrompido, mercado em que, nem sempre, aquele que detém o conhecimento e a técnica é o vencedor, e, sim, aquele que tem o menor preço e fornece as melhores "vantagens", aplicando a velha lei de Gerson.

No capítulo quatro, de um modo geral, busca-se enfocar, principalmente, o panorama das empresas brasileiras que estão procurando superar os atrasos, resquícios da Reserva de Mercado para a Informática, buscando principalmente a certificação de qualidade de seus produtos, na busca de mercados internacionais. Primordialmente será fundamentado na listagem e estudo de procedimentos de melhoria de processos e produtos de softwares, analisando os pontos críticos pertinentes a um software educacional, notadamente referente à interface homem-máquina.

No capítulo cinco, são apresentados os modelos de processo de desenvolvimento e avaliação de software educacional existentes no mercado nacional.

Como resultado desse estudo constatou-se a grande preocupação com o modelo pedagógico adotado e pouca relevância ao modelo de desenvolvimento, aos procedimentos metodológicos ou mesmo aos paradigmas de construção de software. Dessa forma, propõe-se um modelo de desenvolvimento baseado em filosofias e tendências de modelos educacionais, usando como fundamentação as normas e padrões internacionais da ISO e ferramentas sobre engenharia de software; contribuindo para a melhoria do processo ensino-aprendizagem; enfocando a interdisciplinaridade e a multidisciplinaridade, valendo-se de ferramentas de tecnologia da informação.

No capítulo seis, propõe-se a metodologia que baliza todo esse contexto, à luz de critérios. Identificam-se os critérios diretamente relacionados aos aspectos técnicos de construção (*design* e realização), com base no elenco de características das normas ISO, envolvendo equipe multidisciplinar para produção de software, numa perspectiva pedagógica, constituindo-se em suporte para a aprendizagem e recurso para o professor. São consideradas as perspectivas para trabalhos futuros, baseados nas características listadas nesta proposta, como a definição de subcaracterísticas e a especialização de atributos para cada critério proposto.

Constata-se, pela experiência vivida em mais de vinte anos em desenvolvimento de software, que as empresas desenvolvedoras de software estão

solucionando a tal *"crise do software"*, em que convivem a baixa qualidade, prazos vencidos, orçamentos ultrapassados, usuários insatisfeitos e métodos gerenciais empíricos, buscando ciência de métodos e normas de qualidade, adequando-os e incorporando-os às rotinas diárias de desenvolvimento e gerenciamento.

Sobre isso, cita Martin (1991): "o *The Wall Street Journal lamentou que o software seja um dos obstáculos do progresso econômico. Uma autoridade do Pentágono comentou: se alguma coisa nos matar antes dos russos, esta coisa será o nosso software"*.

2. PARAMETRIZAÇÃO NUMÉRICA DO MERCADO DE SOFTWARE

Esse capítulo parametriza em números, demonstrado por meio de tabelas, o mercado do software brasileiro e também uma amostra do mercado americano, como uma comparação com o mercado global, considerando que o americano seja a representação do mercado internacional. Detalha o mercado atual, em forma de pesquisa desenvolvida desde 1993, com periodicidade bienal, pelo Ministério da Ciência e Tecnologia, através da Secretaria de Política de Informática e Automação-SEPIN, no momento, o documento mais abrangente sobre a evolução da indústria de software em nosso país. É uma permanente fonte de referência, indispensável para o estabelecimento de padrões de qualidade e produtividade compatíveis com as exigências internacionais. Dentro do escopo dessa pesquisa, em que se procurou traçar um perfil das empresas brasileiras produtoras de software, está a importância da elaboração de um instrumento para a disseminação dos conceitos de qualidade e produtividade e um dos primeiros indicadores globais, sobre produtividade sistêmica de um setor específico da cadeia econômica do país. Tornou-se essencial a sua inserção, neste ponto, para o compartilhamento dos resultados apresentados, transformando as informações em conhecimento do mercado e evolução da indústria de software, procurando destacar as atuações na área educacional. Destacaram-se também alguns parâmetros sobre a pesquisa do MCT/SEPIN, acrescentando dados da pesquisa realizada pelo CTI/IC-Fundação Centro Tecnológico para Informática/Instituto de Computação de Campinas (São Paulo), sobre Gerência de Configuração de Software em 1999, em que foram processados dados de 355 empresas, com atividades voltadas exclusivamente ao desenvolvimento de softwares.

Para ilustrar o mercado internacional, elegeu-se a pesquisa realizada em 1995 pelo Standish Group Internacional, uma empresa especializada em pesquisa de mercado e consultoria de mercado em software de missão crítica e *e-commerce*. Participaram dessa pesquisa 365 companhias desenvolvedoras de software, representando 8380 aplicações de missão crítica, visando identificar o âmbito das falhas de projetos de software, os fatores principais que causam falhas nos projetos de software e os ingredientes-chave, que podem reduzir essas falhas, reforçando o esforço desenvolvido para melhorar os processos de software, identificando as falhas com os dados da Força

Aérea Americana, sobre o levantamento da fonte de erro por fase do ciclo de vida do software e dados sobre o custo relativo de reparo, por fase do ciclo de vida, originado dos estudos conjuntos das empresas norte-americanas GTE, TRW e IBM, em meados da década de 1970.

2.1 - MERCADO ATUAL DO SOFTWARE BRASILEIRO

Na apresentação do resultado da pesquisa do MCT/SEPIN do ano de 1999, publicado em 2000, diz-se que a taxa média anual de crescimento da receita foi de 19% sobre os valores correntes no setor de software, superando o setor de hardware, que cresceu 6% ao ano, no mesmo período. Trata-se um mercado projetado para o ano 2000 de R$ 5,9 bilhões, provenientes das indústrias produtoras de software instaladas no país, acrescido de US$ 1,2 bilhões para importação, resultante da remessa em direitos autorais.

Visando ao aprimoramento da pesquisa, para o ano de 1999, foi alterado o escopo da pesquisa “Qualidade no Setor de Software Brasileiro”, para “Qualidade e Produtividade no Setor de Software Brasileiro”, desse modo, possibilitando a geração de novos indicadores que apontem as tendências do setor. A população-alvo ficou constituída pelas empresas desenvolvedoras de software, quer seja software pacote para comercialização, software sob encomenda para terceiros, softwares embarcados, ou software para Internet, além das empresas distribuidoras ou editoras de softwares de terceiros, perfazendo 446 empresas participantes de um mercado estimado em 2500 empresas em todo o país.

Conforme Daniel Boacnin, presidente da Associação Brasileira das Empresas de Software-ABES, o setor de software será, segundo pesquisas internacionais, responsável pelos maiores índices de crescimento na economia global nos próximos anos, portanto é fundamental que o Brasil participe desse movimento, numa condição de vanguarda. Para isso, é importante que exista uma estratégia para o aumento da competitividade do setor, baseada em qualidade, custos e eficiência de nível internacional. Essa estratégia só pode ser estruturada e implementada se houver um real e preciso levantamento da situação atual vivida pelo setor e uma clara visão dos objetivos a alcançar, a curto, médio e longo prazo, para garantir uma presença forte e contínua nesse mercado crescente.

A importância de uma pesquisa do porte dessa reside nas informações levantadas, que servirão de base para que o governo, a indústria, as academias, tenham dados precisos para fundamentar a linha de atuação que conduza o setor a uma posição de destaque, tanto dentro do país quanto internacionalmente. O grande desafio é a participação ativa na nova "economia digital", em que o setor de software desponta como agente crítico da atuação brasileira nesse mercado globalizado.

TABELA 01 - PRODUTIVIDADE DE SOFTWARE POR EMPREGADO – 1998

		Porte(comercialização bruta anual de software)			
Porte (nº de pessoas)	**Categorias**	**Micro**	**Pequenas**	**Médias**	**Grandes**
		Até R$120 mil	De R$ 120 a R$720	De R$720 a R$2500	Mais de R$2500
Micro (de 1 a 10)	Nº empresas	79	53	7	-
	Produtividade	13,3	53,1	177,3	-
Pequenas (de 11 a 50)	Nº empresas	15	50	30	18
	Produtividade	3,1	18,1	50,7	337,9
Médias (de 51 a 100)	Nº empresas	1	3	8	13
	Produtividade	X	X	23,8	65,4
Grandes (mais de 100)	Nº empresas	-	4	5	35
	Produtividade	-	X	x	46,4

NOTA: A produtividade de software por empregado refere-se ao valor bruto proveniente da comercialização de software sobre a força de trabalho efetiva nas empresas, medido em R$ mil.

FONTE: MCT/SEPIN-Ministério da Ciência e tecnologia/Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000

Pode-se destacar a participação de micro e pequenas empresas como a grande força de trabalho na indústria de software do país.

Na tabela 02, procura-se demonstrar os principais tipos de softwares desenvolvidos pelas empresas pesquisadas, para destacar a atuação em softwares educacionais, com 44 empresas, numa participação de mercado de 10,4%. Sendo uma questão de múltipla escolha, este percentual não representa o mercado real e, sim, a relação entre o número total de empresas participantes e o número de empresas que desenvolvem softwares educacionais.

Em comparação com os softwares administrativos/financeiros, o resultado torna-se insignificante, apesar de figurarem entre os vinte tipos de softwares mais

desenvolvidos pelas empresas. Segundo uma pesquisa realizada pela Universidade Federal de Juiz de Fora, pelas professoras Fernanda Campos e Fátima Gaio, sobre o perfil do mercado brasileiro de software educacional, a maioria das empresas são de pequeno porte, com poucos produtos no mercado. Em geral, têm boa interação com universidades e centros de pesquisa, buscando, nessas instituições, consultoria e atualização. Há o reconhecimento de que existe pouca mão-de-obra especializada disponível no mercado. A maioria acredita ser este um nicho de mercado que deve ser explorado e não teme a concorrência externa. Todos divulgam seus produtos em feiras, seminários e congressos. As autoras concluem que a educação vem sendo considerada uma justificativa importante para a aquisição de computadores pelos pais; e a escola tem-se caracterizado como a Segunda Onda de compradores dessa tecnologia. Estes são alguns fatores facilitadores da expansão de mercado e as famílias vêem assim a possibilidade de completar a educação com os *softwares* com vida mais longa que a dos vídeo games.

TABELA 02 - PRINCIPAIS TIPOS DE SOFTWARES DESENVOLVIDOS

Categorias	Nº empresas	%
Financeiro	148	34,8
Administração	132	31,1
Contabilidade	107	25,2
Automação comercial	110	25,9
Administração de recursos humanos	86	20,2
Página WEB	81	19,1
Automação industrial	63	14,8
Automação de escritórios	63	14,8
Gestão integrada - ERP	80	18,8
Saúde	53	12,5
Comunicação de dados	42	9,9
Administração pública	70	16,5
Administração escolar	54	12,7
Comércio eletrônico	49	11,5
Administração de serviços	69	16,2
Educacional	**44**	**10,4**
BASE	**425**	**100,0**

NOTA: Questão de múltipla escolha.
FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 03. Brasília: 2000

Na tabela 03, demonstra-se o panorama das empresas de software, em relação ao total de empresas certificadas até o ano de 1997. O diferencial desta amostra está na

informação prestada pela ABNT-CB25 – Associação Brasileira de Normas Técnicas-Comissão Brasileira de Qualidade, sobre o número de empresas certificadas pela norma ISO 9000, de todos os setores, em comparação com a área de Informática diferenciada em dois níveis: ISO 9001 e ISO 9002. A pesquisa mostrou também o crescimento no número de empresas que adotaram um programa de qualidade total, sistemas de qualidade ou similares, índice apontado por mais de 100 empresas, podendo-se observar um número crescente a cada ano, de tal modo que mais da metade foram implantados a partir de 1997.

TABELA 03 - CERTIFICAÇÃO DO SISTEMA DE QUALIDADE	
Empresas Certificadas	**1997**
Todos os setores	1788
Setor de Informática	129
Pesquisa da Qualidade em Software	45
Certificação ISO 9001	36
Certificação ISO 9002	11
SW explicitado no escopo certificado	16

FONTE: ABNT/CB-25 - Associação Brasileira de Normas Técnicas-Comissão Brasileira de Qualidade.

A tabela 04 retrata a evolução das indústrias de software, desde 1993, no uso de sistemas da qualidade, para o aprimoramento da gestão de sistemas e softwares. Nota-se que, a partir de 1997, as empresas desenvolveram uma cultura voltada para a gestão da qualidade, tomando as decisões baseadas em fatos e dados e não mais empiricamente.

TABELA 04 – CERTIFICAÇÃO DO SISTEMA DA QUALIDADE			
	1993	**1997**	**1999**
TODOS OS SETORES		1788	
SETOR DE INFORMÁTICA		129	268
PESQUISA DA QUALIDADE EM SOFTWARE	35	45	74
CERTIFICAÇÃO ISO 9001	3	36	63
CERTIFICAÇÃO ISO 9002	-	11	16
CERTIFICAÇÃO CMM	-		5
SW EXPLICITADO NO ESCOPO DO CERTIFICADO	-	16	39
OUTRAS CERTIFICAÇÕES	12	-	-
FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 03. Brasília: 2000			

Uma pesquisa como essa, realizada periodicamente, serve como um instrumento balizador de tendências e indicador de estágio tecnológico para aqueles que buscam a garantia de qualidade de produtos e serviços, marcando um lugar na corrida pela competitividade, no mundo globalizado, não só em termos de inovação tecnológica, mas na capacidade gerencial das empresas, na promoção da competição, de forma agressiva, em termos de qualidade e produtividade de seus produtos e serviços.

Em uma outra pesquisa realizada pela Fundação Centro Tecnológico para Informática-CTI/IC-Instituto de Computação de Campinas (São Paulo), com o apoio da SEPIN, como um aprofundamento dos dados levantados pelo Programa Brasileiro da Qualidade e Produtividade, abrangeu a Gerência de Configuração de Software-GCS, que visa garantir a integridade dos produtos de software, por meio de Controle de Processo de desenvolvimento que se tem mostrado como um caminho eficaz para se atingir metas. A GCS é um dos processos estabelecidos como requisito básico para a obtenção de certificado de qualidade de processos de software pelas iniciativas internacionais de avaliação e de certificação de qualidade, entre elas a ISO 9000 e CMM. Entre os objetivos dessa pesquisa figuram: o levantamento do perfil de utilização de técnicas de GCS e, num nível mais amplo, a sintonia da atuação do CTI/IC com as necessidades das empresas brasileiras produtoras de software. Participaram da pesquisa 355 empresas, distribuídas por todo o país, que desenvolvem softwares, comercializam pacotes, desenvolvem, por encomenda e para uso próprio, um perfil já caracterizado pela pesquisa do MCT, no âmbito do Subprograma Brasileiro da Qualidade.

Essa pesquisa não incluiu a medição da qualidade nos produtos desenvolvidos pelas empresas. As questões levantadas procuraram identificar a existência de alguma indicação quanto à preocupação da empresa em desenvolver produtos com qualidade, usando ou não algum padrão de qualidade.

TABELA 05 - DISTRIBUIÇÃO DAS EMPRESAS QUANTO A CERTIFICAÇÃO ISO 9001 ENGLOBANDO A ÁREA DE DESENVOLVIMENTO DE SOFTWARE

CERTIFICAÇÃO ISO 9001	Nº EMPRESAS	%
Já possui a certificação	28	7,9
Não tem conhecimento	25	7,1
Não possui a certificação, mas está interessada	233	66,2
Não possui a certificação e não está interessada	66	18,8
Total	**352**	**100**

FONTE: CTI/PQPS-Centro Tecnológico para Informática/Programa de Qualidade e Produtividade em Software. Campinas-SP. 1999.

Nas duas amostragens, nota-se a preocupação da empresas em produzir softwares com qualidade para atender a demanda cada vez maior de produtos de software que atendam à exigência de qualidade internacional, competindo em igualdade de condições, aplicando os padrões de qualidade e produtividade adotados pelo mercado global. O número de empresas certificadas cresce a cada nova edição da pesquisa, conforme demonstram as tabelas.

A tabela a seguir demonstra o nível de conhecimento e o uso efetivo de normas e modelos para a construção de indicadores de qualidade dos processos de software e da qualidade dos produtos de software. São focados os modelos e normas que dizem respeito à gestão de softwares e sistemas como a norma ISO/IEC 12207, Information Technology - Software Life Cycle Process, que define os processos de software, o Capability Maturity Model-CMM, modelo para avaliação da maturidade de processos de software de uma organização e para a identificação das práticas-chave requeridas para aumentar a maturidade desses processos, o projeto Software Process Improvement and Capability Determination-SPICE, a qual, em 1993, foi estabelecido pela ISO, como a futura Norma ISO/IEC 15504, a norma ISO/IEC 12119, que trata de testes e requisitos de qualidade para pacotes de software e a norma ISO/IEC 9126, para a avaliação de produto de software. Essas são algumas das normas e modelos de sistemas da qualidade abordados na pesquisa, para situar as empresas produtoras de software em um ambiente de evolução e inovação em gestão tecnológica.

TABELA 06 - Conhecimento dos modelos e normas de qualidade

Categorias	**CMM**		**SPICE**		**ISO/IEC 12207**		**12119**	**9126**
	Nº	**%**	**Nº**	**%**	Nº	%	Nº	Nº
Conhece e usa	8	1,8	1	0,2	16	3,6	4	15
Conhece e começa a usar	36	8,1	14	3,2	53	11,9	23	27
Conhece, mas não usa	165	37,2	121	27,3	121	27,2	114	119
Nâo conhece	234	52,8	308	69,4	255	57,3	304	284
BASE	**443**	**100**	**444**	**100**	445	100	445	445

FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/ Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000

Quanto ao processo de desenvolvimento de software, foram abordados os métodos de Engenharia de Software adotados pelas empresas, as ferramentas de

desenvolvimento utilizadas, o uso de métricas e a documentação adotada, compondo a categoria quatro dessa pesquisa do MCT, dentro do Subprograma de Qualidade e Produtividade de Software, em 1999. Para efeito dessa pesquisa, os métodos de Engenharia de Software foram agrupados em dois subconjuntos distintos, voltados para a melhoria da qualidade de software, de acordo com a classificação proposta por Caper Jones: métodos para prevenção de defeitos e métodos para detecção/remoção de defeitos.

Constatou-se que os principais métodos utilizados pelas empresas, para prevenção de defeitos foram: adoção de normas e padrões da própria empresa, prototipação, análise crítica conjunta e gerência de projetos; as empresas que não adotam alguma metodologia, representam um percentual muito baixo, conforme tabela abaixo.

TABELA 07 - MÉTODOS DE ENGENHARIA DE SOFTWARE P/PREVENÇÃO DE DEFEITOS

CATEGORIAS	Nº EMPRESAS	%
Normas e padrões da empresa	226	53,1
Prototipação	187	43,9
Análise crítica conjunta	183	43
Gerência de Projetos	180	42,3
Verificação	155	36,4
Processo de sw definido e documentado	133	31,2
Reuso	104	24,4
Auditorias	88	20,7
Gestão de Configuração	63	14,8
Medições da qualidade (métricas)	52	12,2
Normas e padrões internacionais	45	10,6
Não adota métodos de prevenção	41	9,6
Base	**426**	**100**

FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000.

Quanto ao grupo dos métodos adotados para detecção de defeitos, apresentou, conforme mostra a tabela 08, um percentual maior do que a tabela anterior de prevenção de defeitos.

Dentre os métodos para detecção de defeitos, foram considerados avançados, pelos analistas da pesquisa, os testes de aceitação com 48,1% e a de validação com 45,1%. A inspeção formal considerada como “avançada” foi assinalada por 20,4% dos pesquisados, o que, na opinião dos analistas, ainda não se configura como um percentual de destaque.

TABELA 08 - MÉTODOS DE ENGENHARIA DE SOFTWARE P/DETECÇÃO DE DEFEITOS		
CATEGORIAS	**Nº EMPRESAS**	**%**
Testes funcionais	263	61,7
Testes de campo	255	59,9
Testes de aceitação	205	48,1
Testes do sistema integrado	199	46,7
Validação	192	45,1
Testes de integração	190	44,6
Avaliação do produto	179	42
Avaliação da usabilidade	149	35
Documentação formal de teste	115	27
Planejamento formal de testes	107	25,1
Inspeções formais	87	20,4
Não adota métodos para detecção	31	7,3
Base	**426**	**100**

FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automacão. Oualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000.

Adicionalmente, foram pesquisadas outras práticas de Engenharia de Software com grande destaque, como: o controle de versão, modelagem de dados, projeto de interface com o usuário, dicionário de dados e métodos orientados a objeto; os considerados "avançados" são o projeto de interface com o usuário, métodos orientados a objeto, métodos estruturados e gestão de mudança.

Quanto à aplicação de indicadores de qualidade e produtividade que permitem avaliar os processos de software, foram assinaladas o uso de métricas, tais como: linhas de código (LOC) e pontos por função (*function point*, mais conhecida como FPA).

TABELA 09 - MÉTRICAS USADAS P/MEDIÇÃO DA QUALIDADE DE PROCESSOS DE SW		
CATEGORIAS	**Nº EMPRESAS**	**%**
Linhas de código (LOC)	57	12,8
Pontos por função (FPA)	85	19,1
Outros	88	19,8
Não utiliza	242	54,4
Base	**445**	**100**

FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automação. Qualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000.

A medição de linhas de código foi a métrica mais aplicada no passado, quando o código era dominante nas estimativas de custo e produção de software. Desde a década de 1990, vem ganhando espaço a técnica de pontos por função, ou seja, uma técnica de avaliação e medição de software, baseada na medição do valor das funções

executadas pelos programas, em vez de utilizar como base o volume ou a complexidade do código dos programas, mais conhecida como FPA-*Function Point Analysis*.

Considerando que, na pesquisa realizada em 1997, as empresas que usavam a técnica FPA para medição da qualidade e produtividade de processos de software foi de 4,4% e as que estavam em estudo ou implantando, em torno de 16%, constata-se que as empresas estão investindo em capacitação de pessoal e em métodos e técnicas para estar à altura do mercado global.

TABELA 10 - MÉTRICAS PARA MEDIÇÃO DA PRODUTIVIDADE DE PROCESSOS DE SW

CATEGORIAS	**Nº EMPRESAS**	**%**
Linhas de código (LOC)	61	13,7
Pontos por função (FPA)	84	18,8
Outros	45	10,1
Não utiliza	275	61,7
Base	**446**	**100**

FONTE: SEPIN/MCT-Ministério da Ciência e Tecnologia/Secretaria de Política de Informática e Automacão. Oualidade no Setor de Software Brasileiro - 1999. Nº 3. Brasília: 2000.

2.2 PANORAMA INTERNACIONAL

Para ilustrar o mercado internacional, elegeu-se a pesquisa realizada em 1995, pelo Standish Group Internacional, uma empresa especializada em pesquisa de mercado e consultoria de mercado em software de missão crítica e *e-commerce*. Participaram dessa pesquisa 365 companhias desenvolvedoras de software, representando 8380 aplicações de missão crítica, visando identificar o âmbito das falhas de projetos de software, os fatores principais que causam falhas nos projetos de software e os ingredientes chave que podem reduzir essas falhas, reforçando o esforço desenvolvido para melhorar os processos de software, identificando as falhas, com os dados da Força Aérea Americana sobre o levantamento da fonte de erro por fase do ciclo de vida do software e dados do custo relativo de reparo por fase do ciclo de vida originado dos estudos conjuntos das empresas norte-americanas GTE, TRW e IBM em meados da década de 1970.

O estudo mostra que a identificação de um erro, na fase de manutenção, tem um custo relativo 200 vezes maior em relação à descoberta do mesmo erro, na fase de análise de requisitos, na fase inicial do ciclo de vida; a maior fatia da amostra incide sobre erros de requisitos em 41%, erros de design lógico em 28% e outros erros como dados, ambiente, documentação, interface, são erros com percentual abaixo de 5%.

Nesse sentido, o esforço de melhoramento deve ser executado na fase inicial, quando se definem os requisitos do usuário e a interação é muito forte, como forma de aumentarmos a qualidade do software, bem como de minimizar os custos de desenvolvimento.

De acordo com a pesquisa:

- os Estados Unidos gastam US$ 250 bilhões por ano em aproximadamente 175 mil projetos;
- de todos os projetos, 31% são cancelados antes do término, representando um desperdício da ordem de US$ 81 bilhões;
- de todos os projetos, 53% chegam ao final, tendo custado 189% do valor estimado, representando US$ 59 bilhões em custo adicional, atrasam em até 222% da estimativa original, além de serem entregues com apenas 61% das características originalmente especificadas;
- do total, 16% são entregues no prazo e dentro do orçamento.

O aspecto mais importante da pesquisa está no levantamento dos fatores principais que tornam um projeto crítico, ou seja, aqueles que extrapolam prazo, custo e que são entregues com a funcionalidade prejudicada.

Foram identificados fatores como:

FATORES CRÍTICOS DE PROJETO	Nº RESPOSTAS
Falta de Especificação do usuário	12,8%
Requisitos Incompletos	12,3%
Mudança de requisitos	11,8%
Falta de apoio executivo	7,5%
Tecnologia imatura	7,0%
Falta de recursos	6,4%
Expectativas irreais	5,9%
Objetivos obscuros	5,3%
Tempo irreal	4,3%
Tecnologia nova	3,7%
Outros	23,0%

Pela pesquisa, demonstra-se que os desenvolvedores não se comunicam na mesma linguagem que os usuários. Está clara a inabilidade dos que desenvolvem o

projeto, para trabalhar mais efetivamente com o usuário e entender melhor as suas necessidades (requisitos) e expectativas (objetivos).

2.3 CONCLUSÃO DO CAPÍTULO

Conforme Kival Chaves Weber (1994), diretor presidente da Sociedade SOFTEX-Sociedade Brasileira para Apoio à Exportação de Software, *para competir no mercado global, além dos fatores custo e inovação, é necessário atender padrões internacionais de qualidade...* E isso está sendo seguido ao "pé da letra", como uma "lição de casa" pelas empresas brasileiras, buscando continuamente o aprimoramento dos processos de desenvolvimento de software, baseando-se em fatos e dados embasados em pesquisas sérias, formando um banco de informações fidedignas, sobre a qualidade do software produzido no país.

Em comparação ao mercado global, os problemas e as soluções se constituem os mesmos, tanto no Brasil como em mercados internacionais, conforme demonstram as informações resultantes das pesquisas analisadas neste capítulo.

No item referente aos tipos de software desenvolvidos pelas empresas (Tabela 02), de um resultado global de 10,4%, as empresas desenvolvedoras de softwares educacionais dividem-se em empresas que desenvolvem pacotes de software (12,6%), empresas que desenvolvem sob encomenda (10,1%), empresas que desenvolvem softwares embarcados, empresas que desenvolvem softwares educacionais para Internet (21,2%), empresas que desenvolvem softwares para uso próprio (14,9%), empresas que distribuem softwares educacionais (14,0%) e o restante constituído de empresas que, de alguma forma, trabalham com softwares educacionais, mas que não se enquadram em nenhuma das categorias citadas. Ainda que seja um mercado pouco explorado, conforme dados da pesquisa de 1999, está entre os 20 tipos mais desenvolvidos pelas empresas. Este resultado não difere muito da pesquisa de 1997, que foi de 9,5% do total de empresas pesquisadas, mostrando uma estabilidade pouco saudável e ainda incipiente para o atendimento do mercado educacional.

É fundamental que o Brasil participe do movimento de globalização da economia mundial e participe com uma fatia representativa neste mercado. Para isto, é importante que exista uma estratégia para o aumento da competitividade do setor de software, baseada em qualidade, custos e eficiência de nível internacional. Essa

estratégia só pode ser estruturada e implementada, se houver um real e preciso levantamento da situação atual vivida pelo setor e uma clara visão dos objetivos a alcançar, a curto, médio e longo prazo, para garantir uma presença forte e contínua nesse mercado crescente.

A importância das pesquisas de mercado reside nas informações levantadas, as quais servirão de base para que o governo, a indústria, as academias, tenham dados precisos para fundamentar a linha de atuação, que conduza os rumos do setor e formule estratégias para o amadurecimento de tecnologias e reduza a complexidade dos problemas.

O grande desafio é a participação ativa na nova “economia digital”, em que o setor de software desponta como agente crítico da atuação brasileira no mercado globalizado.

3 – MÉTODOS E TÉCNICAS DE DESENVOLVIMENTO DE SOFTWARE

Desde que Thomas J. Watson, Sr. Fundador da IBM, disse em 1943: *Acho que há um mercado internacional para cerca de cinco computadores***,** e Ken Olsen, fundador da DEC, em 1977, disse: *Não há razão para que um pessoa tenha um computador em sua casa***,** podemos dizer que a história caminha a passos largos (citados por Weinberg , 1993).

A "idade" deste desenvolvimento, pode-se dizer que beira os 50 anos, sempre fundada em processos empíricos, fortemente baseado em valores individuais e no uso de ferramentas artesanais. O produto de software sempre foi considerado uma arte, em que todo o funcionamento do sistema estava calcado no artista, ao qual pertencia o dom de resolver problemas e a quem todos deviam veneração, pela sua capacidade e pelo fruto da inspiração e intimidade entre ele, o profissional, e o seu computador.

Esta veneração atenuou-se, quando surgiram os primeiros compiladores ou linguagens de programação em alto nível. Com a multiprogramação, partindo de uma situação em que se tinha um único programa na memória do computador, para "n" programas carregado em "n" partições da memória, conseguiu-se obter ganho em rendimento de processamento de programas. Quando surgiram os Banco de Dados, entendidos como integração de dados, acompanharam os gerenciadores destes bancos, trazendo novas técnicas de programação e estruturação de dados.

Quando surgiu o conceito de "Memória Virtual", com as máquinas IBM, as ferramentas de desenvolvimento de softwares foram tomando a forma de "Engenharia de Software", isto é, os conceitos de programação estruturada tornaram-se ferramentas úteis na obtenção de produtividade e melhoria do desenvolvimento de softwares.

Levados pela necessidade, pela posição estratégica do software na humanidade em atividades críticas, os primeiros passos em direção à profissionalização pela indústria do software foram dados pelo surgimento de novas técnicas e metodologias de desenvolvimento. Com base nestas, surgiram novas linguagens de desenvolvimento e gerenciamento, baseadas em processos e produtos.

Na década de 60, com a crescente especialização na área de desenvolvimento de softwares, puderam determinar que essa atividade poderia ser dividida em fases como: Análise, Projeto, Implementação, Integração, Teste e Manutenção, constituindo

todo o ciclo de vida do sistema.

Todo e qualquer software é desenvolvido, independentemente de plataforma e tecnologia, segundo uma seqüência básica de grandes atividades, denominada Ciclo de Vida.

Maffeo (1992) afirma que:

> **"desenvolvimento do software ocorre no contexto de um projeto, que é executado de acordo com um processo, gerando produtos intermediários e um produto final - *release* do software -, que deve ser mantido a fim de corrigir defeitos introduzidos durante o projeto, ser melhorado com a incorporação de novas *features* e por fim, ser desativado, substituído ou descartado, ao final de sua vida útil".**

Podemos dizer que a história da engenharia de software é a história baseada em tentativas de superar a dinâmica tamanho/complexidade criando simplificações criativas visando reduzir a complexidade de soluções, à medida que os problemas ficam maiores. A engenharia de software baseia-se na necessidade de ambição, pelas soluções simples para problemas complexos. (WEINBERG, 1994).

Para tratar o software sob uma abordagem de engenharia é necessário entendermos as características de sua dimensão e a importância da medição para a sua gestão, conforme Fernandes (1995), a *gestão do software consiste nas ações necessárias para a gestão do projeto, do ambiente de desenvolvimento e do produto (release), visando a atingir objetivos previamente estabelecidos no que concerne a produtividade e qualidade e quanto ao atendimento das necessidades dos usuários (clientes).*

3.1 DEFINIÇÕES

3.1.1 Projeto de Software

Um projeto de software é um esforço no sentido de construir um produto, dentro de determinadas especificações, que atenda às necessidades dos usuários para que executem processos operacionais e gerenciais de negócios. É a junção de Objetivos + Atividades + Prazos + Recursos Envolvidos + Riscos e Incertezas, conforme Fernandes (1995).

É o início do trabalho de detalhamento, a que se devem incorporar as

características da tecnologia de automação a ser empregada; seu produto devendo ser validado em relação ao resultado da atividade de análise de requisitos ou à especificação das necessidades. (MAFFEO, 1992).

É uma fase relativa à forma pela qual os processos selecionados são implementados em procedimentos e como estes funcionam. Faz-se necessário um envolvimento direto do usuário final no projeto dos procedimentos e na interação com os protótipos. (MARTIN, 1991).

É a fase em que comprador e fornecedor concordam, previamente, sobre o conjunto de informações do fornecedor, que serão repassadas ao comprador. O desenvolvedor deve usar uma metodologia sistemática e, dentro do possível, valer-se da experiência passada. O projeto deve levar em consideração as futuras atividades de manutenção e aderir às regras e convenções de programação, conforme a ISO-9000-3.

Como exemplos de projetos de softwares, podemos citar: o desenvolvimento de produto de software, desenvolvimento de softwares específicos e personalizados, desenvolvimento de reengenharia e reconstrução, culminando em novas versões de software. Os serviços de manutenção, adaptações à legislação, pequenas alterações e pequenos programas incorporados ao projeto não são considerados como projeto de software.

3.1.2 Processo de Software

O processo de software é a forma como o projeto será implementado e executado. Podemos dizer que o processo constitui-se na arquitetura do software. São geralmente denominados de *"programas"*, *"co-rotinas"*, *"task"* e outros.

O processo de software é um conjunto de atividades (numa seqüência pré-determinada), métodos e práticas utilizados na produção e evolução do software, conforme Fernandes (1995).

Os processos de software são unidades de execução de software. Genericamente, entende-se por processo qualquer conjunto de ações que possa ser referenciado globalmente pelo ambiente operacional de um dado processador, segundo a concepção de Maffeo (1992).

Pela Norma ISO 8402, processo é o conjunto de recursos e atividades inter-relacionados, que transformam insumos em produtos.

3.1.3 Produto de Software

A Norma ISO 8402 (ABNT, 1993) diz que produto é o resultado de atividades ou processos. Inclui serviços, materiais e equipamentos, materiais processados, informações ou uma combinação destes. Um produto pode ser tangível (materiais processados) ou intangível (informações e conceitos), ou uma combinação destes. Um produto pode ser intencional (oferta aos clientes) ou não intencional (um poluente ou efeitos indesejáveis).

A Norma ISO/IEC 9126 diz que um produto de software é uma entidade de software disponível para liberação a um usuário. Um software: programas, procedimentos, regras e qualquer documentação associada, pertinente à operação de um sistema computacional.

3.2 METODOLOGIAS E TÉCNICAS

3.2.1 Metodologia de Projeto Warnier-Orr

A metodologia Warnier-Orr usa a teoria dos conjuntos para descrever projeto de programas. Um conjunto é uma coleção ordenada de objetos que compartilham uma ou mais características comuns.

Etapas do processo de Projeto de Warnier-Orr:

a) definir as saídas dos processos: cada saída do programa é uma estrutura de dados hierárquica;
b) definir o Banco de dados lógico: definir todos os elementos de dados para produzir as saídas do programa;
c) executar a análise de eventos: definir todos os eventos que possam afetar os elementos de dados do Banco de Dados lógico;
d) desenvolver o Banco de Dados físico: definir os arquivos físicos para os dados de entrada;
e) projetar o processo lógico: projetar a lógica do processamento do programa, necessária para produzir a saída desejada a partir da entrada, e
f) projetar o processo físico: acrescentar a lógica de controle e os procedimentos de tratamento de arquivos, para completar o projeto do programa.

Essa metodologia está baseada em abordagens orientadas para os dados e produzem a estrutura de programa, a partir da estrutura de dados. Produz, também, a estrutura de programa e as estruturas de dados de entrada, a partir das estruturas de dados de saída. A filosofia desta metodologia é que a saída do programa determina completa e absolutamente a estrutura de dados, que, por sua vez, determina a estrutura de programa, sendo chamada de *"análise orientada para a saída"*.

3.2.2 Metodologia de Projeto Jackson

Esta é uma técnica de projeto de programa, orientada para dados. Sua abordagem é derivar a estrutura de programa, a partir das estruturas de dados. Supõe que o problema foi totalmente especificado e que o programa será implementado com uma linguagem de programação procedural, de segunda ou terceira geração. Portanto, a preocupação com a análise de sistemas e a implementação de programas fica fora do processo de projeto.

Jackson vê um programa como um processo seqüencial, que tem entrada e saídas que são vistas como fluxos seqüenciais de registros. Recomenda que pensemos em cada fluxo de dados como um arquivo em fitas, para reforçar a idéia de separar os programas e limitar a comunicação a um protocolo simples e seqüencial. O processo de projeto consiste em: primeiro, definir a estrutura dos fluxos de dados e, então, organizar a lógica funcional para fazer as estruturas de dados.

3.2.3 Ferramentas Case

Várias ferramentas CASE para análises gerais e de projeto evoluíram a partir da metodologia de Análise Estruturada, como as Metodologias de Jackson e Warnier. As técnicas estruturadas, em geral, dão destaque aos projetos diagramáticos e esquemáticos, o que facilita as comunicações entre a organização que redige a especificação dos requisitos e a equipe de desenvolvimento de software.

Essas ferramentas pertencem à área de Engenharia de Sistemas, Computer Aided Software Engineering-CASE, que veio com a intenção de auxiliar engenheiros de softwares e os profissionais que atuam na área a especificar e projetar softwares.

CASE possui algumas características essenciais que ajudam a trabalhar nas atividades de especificação, projeto, implementação, teste e documentação de softwares:

a) geração automática de código, a partir da especificação;
b) ferramentas integradas ao software de apoio;
c) prototipação;
d) ferramentas embutidas, proporcionando padronização do uso de metodologias;
e) banco de dados do projeto;
f) gráficos com diagramas estruturados;
g) cobertura de todo o ciclo de vida do software.

As ferramentas CASE são utilizadas para fortalecer as duas principais fases dentro do desenvolvimento de software: a análise dos requisitos e a especificação do projeto. Essas fases são, nada mais nada menos, que o desenvolvimento de software seguido da implementação e manutenção; a ferramenta CASE dá cobertura às duas primeiras fases, e as demais são obtidas por compiladores e depuradores, que são ferramentas atuais.

No processo de seleção de ferramentas CASE, é indispensável definir uma lista de características que devem ser analisadas, considerando os aspectos que sejam importantes para a adoção de um critério de seleção. Vários pesquisadores têm voltado suas pesquisas à análise dos aspectos humanos, sociais, organizacionais e políticos da implantação de ferramentas e não apenas aos aspectos técnicos.

3.2.4 Análise Orientada a Objetos

Com o barateamento do hardware e a grande evolução em linguagens de programação, baseados em Banco de Dados, paralelamente à discussão das ferramentas CASE, começa a surgir a filosofia da “Orientação a Objeto”. A primeira vez que se discutiu programação baseada em objetos foi no final dos anos 70, por aqueles que trabalhavam com a linguagem SIMULA; mais tarde, veio o projeto orientado a objeto e também análise orientada a objeto. Basicamente são ferramentas de interfaces gráficas com o usuário.

Tanto as propostas das metodologias estruturadas da década de 70 como as metodologias orientadas a objeto da década de 80 apontam: passos seqüenciais e lógicos, a seguir, documentação dos produtos e estabelecimento de revisões. O problema é que a teoria é muito diferente da prática das organizações.

A abordagem OO caracteriza-se por ser uma importante ferramenta para lidar com sistemas complexos, conduzindo para uma subdivisão do espaço de estados de um sistema entre os diferentes objetos que o integram, fazendo uso de hierarquias de classes e de objetos.

A estrutura típica de um sistema OO envolve uma série de entidades, chamadas de objeto, que interagem entre si, a fim de que o sistema tenha o comportamento desejado. A redefinição de uma rotina aliada à conexão dinâmica entre objetos são técnicas de desenvolvimento de softwares flexíveis e a subcontratação de serviços por outros objetos também contribui para a construção de softwares sem erros e robustos.

A desvantagem de OO em comparação com outros métodos está no fato de que está mais voltada para a melhoria da qualidade, sob o aspecto técnico do desenvolvimento e manutenção de software, sem maiores preocupações com a administração e o acompanhamento da qualidade.

3.3 MODELOS E NORMAS DE QUALIDADE

Em qualquer atividade produtiva humana, o problema metodológico é colocado a partir do momento em que surge a necessidade de racionalizar o processo produtivo, visando atender a objetivos organizacionais associados a: padronização, planejamento, controle, produtividade, eficiência, e qualidade. É o resultado do conjunto de experiências de muitos que compartilham uma maneira simplificada e segura de fazer o mesmo trabalho e que proporciona soluções de maior qualidade ao serviço a ser realizado.

A metodologia torna-se um fator de risco, quando as pessoas envolvidas sentem-se ameaçadas pela mudança de ambiente que os cerca, também pela sensação de perda de controle de uma situação já conhecida. Entretanto a evolução e a adoção de novas formas de solucionar problemas, com a implantação de formas organizadas, padronizadas e principalmente documentadas, permite um *feedback* muito importante durante o processo de produção, assegurando a correção do rumo em qualquer tempo, gerando produtos de qualidade, baseada em uma estrutura metodológica consolidada.

Neste trabalho serão apresentados os modelos relacionados abaixo, em que alguns adotam a abordagem de processo, outros de produto:

a) ISO 9000-3, guia para a aplicação da ISO 9001 para o desenvolvimento,

fornecimento e manutenção de software, abordando os processos de produção de software, com o objetivo de fornecer orientação, quando um contrato entre duas partes exigir a demonstração da capacidade de um fornecedor desenvolver, fornecer e manter produtos de software;

b) ISO/IEC 12207, norma para processos de Ciclo de Vida de software, que define a estrutura de processos necessários para o desenvolvimento de software e fornece as diretrizes para embasar o contrato entre duas partes;
c) CMM - *Capability Maturity Model*, modelo proposto pelo Software Engineering Institute, para o Departamento de Defesa dos EUA, que fornece uma estrutura para apoiar as empresas produtoras de software na evolução dos seus processos e ajuda a priorizar esforços de melhoria contínua dentro da organização, baseado em medições da capacidade e maturidade dos processos de desenvolvimento de software;
d) SPICE - Software Process Improvement and Capability determination, resultado do projeto de geração de normas para melhoria e avaliação do processo de software, promovido pela ISO, futura norma ISO 15504, com o objetivo de fornecer uma estrutura para medição da capacidade de processos de desenvolvimento de software com propostas de melhoria contínua;
e) ISO/IEC 9126 ou NBR 13596, um guia para a avaliação de produto de software – características de qualidade e diretrizes para seu uso, tendo como principal foco a definição de características e subcaracterísticas de qualidade do produto de software a ser aplicado ao longo do ciclo de vida de desenvolvimento de software;
f) ISO/IEC 14598, uma série composta de 6 partes, que aborda a avaliação de produto de software;
g) ISO/IEC 12119, norma para avaliação de pacotes de software, abordando testes e requisitos de qualidade;
h) TRILLIUM - modelo de avaliação da organização proposto pela Bell Canadá, na área de telecomunicação, sendo um dos pioneiros na definição de padrões de comportamento sistematizado, e
i) SQUID - *Software Quality In the Development*, proposto por um consórcio europeu.

Este último, o SQUID, adota a abordagem do software como uma visão integrada de produto e processo. Estes modelos são os mais conhecidos e divulgados, portanto, a abordagem e a escolha para relacioná-los para a ciência de seu conteúdo.

3.3.1 ISO 9000-3

É um guia para a aplicação da ISO 9001 para o desenvolvimento, fornecimento e manutenção de software. A norma ISO 9001 faz parte da série de normas ISO 9000, voltadas para a gestão e garantia da qualidade. Esses documentos especificam os requisitos mínimos para que as empresas possam assegurar a qualidade de seus produtos e serviços, não definindo modelos ou impondo sistemas de qualidade a serem implementados nas organizações. As empresas definem seus próprios modelos de gestão da qualidade, dependendo do seu tipo de negócio e suas características.

A norma ISO 9000-3 foi criada em junho de 1993. Ela se espelha nos itens da ISO 9001, fazendo a necessária adaptação. Para cada item da ISO 9001, existe um correspondente na ISO 9000-3, que o detalha e o adequa ao software.

Esta norma se aplica àqueles casos em que existe um contrato formal entre fornecedor e cliente, isto é, aos casos de desenvolvimento, fornecimento e manutenção de software, desde que especialmente orientado para um cliente, caracterizando assim a relação um para um, em que a qualidade, neste contexto, significa a conformidade com as especificações firmadas entre as partes (ANTONIONI E BRAGA, 1995).

As diretrizes da ISO 9000-3 cobrem questões como: entendimento comum entre as partes (contratante e contratada) de requisitos funcionais; o uso de metodologias consistentes para o desenvolvimento de software e gerenciamentos do projeto como um todo (da concepção até a instalação do software no cliente).

A ISO 9000-3 divide-se em três partes principais:

a) estrutura do sistema da qualidade;
b) atividades do ciclo de vida do software;
c) atividades de suporte.

3.3.1.1 Estrutura do sistema da qualidade

Descreve os aspectos organizacionais relacionados ao sistema da qualidade. Descreve as responsabilidades e ações relacionadas à qualidade, que devem ser tomadas

tanto pelo fornecedor (empresa de software) como pelo usuário (cliente). São quatro os pontos cobertos por este capítulo: estabelecimento da responsabilidade da administração, política formal de qualidade (definição e documentação), procedimentos para auditoria interna do sistema da qualidade e procedimentos para ação corretiva.

O que diferencia, em termos de participação do cliente da norma ISO 9001, são as diretrizes da norma ISO 9000-3, que prevêem atribuições também para eles. É proposto que o cliente indique um representante para negociar com o fornecedor de software as questões contratuais, incluindo definição de requisitos, definição de critérios de aceitação e acordos de conclusão. Reuniões de revisão e acompanhamento entre fornecedor e cliente também são propostas, na primeira parte da norma ISO 9000-3, cujas diretrizes se repetem para cada uma das fases do ciclo de vida do software.

3.3.1.2 Atividades do ciclo de vida do software

Esse item da norma descreve as fases do projeto de desenvolvimento de software e a definição do ciclo de vida em fases, dentro destas, as atividades de cada uma delas. A norma define que o desenvolvimento de software deve ser feito segundo um determinado modelo de ciclo de vida, e as atividades relacionadas à qualidade devem ser planejadas e implementadas de acordo com a natureza deste modelo.

Uma estrutura comum das fases do ciclo de vida do software compreende:

a) fase 1 - definição de requisitos;
b) fase 2 - projeto;
c) fase 3 - implementação;
d) fase 4 teste;
e) fase 5 - liberação para produção;
f) fase 6 - liberação para comercialização.

Independentemente do modelo de ciclo de vida adotado pelas empresas, a norma prevê que as atividades do ciclo de vida podem ser agrupadas nas seguintes categorias:

a) análise crítica de contrato: define padrões que devem constar nos contratos de compra e venda, como a abrangência do trabalho, contingências, responsabilidades e proteção de informações proprietárias;
b) especificação de requisitos do comprador: inclui aspectos de performance,

confiabilidade, segurança e privacidade, além de recomendações para cooperação mútua;

c) planejamento do desenvolvimento: inclui a definição do projeto, organização dos recursos, fases, cronograma e planos de teste. Inclui também formas de controle de entrada e saída para cada fase do ciclo de vida e um método de monitoramento e verificação do andamento de cada fase;
d) planejamento da qualidade: aborda a elaboração de um projeto específico para determinado cliente e contrato. Deve tratar dos objetivos da qualidade como: redução do reprocessamento, redução de número de chamadas de assistência técnica, redução do número de horas de treinamento, critérios de saída de cada fase e entrada seguinte, estabelecimento de atividades de verificação e validação, bem como responsabilidades específicas para atividades da qualidade;
e) projeto e implementação: concordância entre fornecedor e cliente sobre o projeto. O uso de metodologia sistemática para desenvolvimento, adesão às regras e convenções de programação; é recomendado o uso de experiências passadas;
f) teste e certificação: define a necessidade de teste e homologação do software em vários níveis. Esses planos de teste devem cobrir: ambiente, documentação, cases de teste e dados e a validação do sistema completo, bem como teste de campo (teste Beta);
g) aceitação: cobre questões como cronograma, teste e critérios de aceitação, procedimentos para avaliação, ambientes e recursos de software e hardware;
h) reprodução, expedição e instalação: registro relativo ao número de cópias, tipo de meio físico utilizado, direitos autorais e licenças de uso, critérios de envio e obrigações do fornecedor e do comprador ligados à instalação;
i) manutenção: define a manutenção como um item da qualidade, em que esse serviço é incluso no contrato de compra, após a entrega e a instalação inicial. Atividades de manutenção cobrem, normalmente, mudanças no software, solução de problemas, correção de defeitos, modificação de interfaces, melhorias de desempenho e expansões funcionais (*upgrades*). A

norma propõe a existência de um plano de manutenção, documentação e critérios de liberação, em função da incorporação de alterações no software.

3.3.1.3 Atividades de suporte do Sistema da Qualidade

Essa parte da norma descreve as atividades que apóiam as atividades do ciclo de vida de desenvolvimento de software.

Estão organizadas em nove itens:

a) gestão de configuração: identificar, de forma inequívoca, cada versão do software (não basta a data; se houver possibilidade de se ter mais de uma versão no mesmo dia), controlar a atualização simultânea do software por mais de uma pessoa, identificar e seguir todas as alterações resultantes de uma solicitação de alteração e outros;

b) controle de documentos: a norma prevê que, no mínimo, os seguintes documentos sejam controlados: descrição do sistema da qualidade, ao longo do ciclo de vida do software; documentação do planejamento de atividades do fornecedor e de suas interações com o cliente; e os documentos relativos ao produto;

c) registros da qualidade: indica que o fornecedor deve manter os registros da qualidade, de forma que sejam prontamente recuperáveis;

d) medição: a norma admite que não há indicadores de qualidade de software aceitos universalmente, no entanto, adverte que deve ser usado algum tipo de indicador para expressar as falhas e/ou defeitos que façam sentido do ponto de vista do cliente e não somente da empresa de software fornecedora. Devem ser mantidas as medições relativas a produtos e processos;

e) regras, práticas e convenções: este item vai à essência da doutrina das normas ISO, ao estipular que cada fornecedor de software (e não a norma) deve definir suas regras, práticas e convenções, desde que tornem efetivo o sistema de qualidade estipulado na ISO 9000-3;

f) ferramentas e técnicas: enfatiza a responsabilidade do fornecedor, a utilização de ferramentas e técnicas que tornem efetivas as diretrizes das normas;

g) aquisição: é de responsabilidade do fornecedor de software a composição

final do produto de software, para garantir a qualidade, desde a contratação de subfornecedores até a validação dos produtos adquiridos;

h) produto para ser incluído no software: trata das integrações a serem feitas (se necessário) ao software contratado, e

i) treinamento: responsabilidade do fornecedor relativa à identificação das necessidades de treinamento interno para qualificação de seu pessoal.

A norma ISO 9000-3 não trata, com ênfase, a melhoria contínua do processo, apenas propõe o controle da não conformidade de um produto de software e recomenda ações corretivas e preventivas, outras, como o CMM, tratam explicitamente deste item.

Esta norma motivou o aparecimento de outras iniciativas em relação à qualidade de software. A procura da certificação para alcançar um mercado globalizado é o grande motivador do movimento em relação à qualidade de desenvolvimento de software, o que também motiva o aparecimento de modelos de avaliação e melhoria dos processos de desenvolvimento e do produto final de software.

3.3.2 ISO/IEC 12207

Desenvolvida durante os últimos seis anos, aprovada recentemente pela Joint Technical Committee 1-JTC1 of the International Organization for Standardization and the International Electrotechnical Commission nos Estados Unidos, foi publicada em 1995 como Norma Internacional. Enquanto o software tem sido estabelecido como uma parte integrante das disciplinas de ciências e de negócios, os ambientes de desenvolvimento e gerenciamento de software têm proliferado sem uma estrutura comum e uniforme para o Ciclo de Vida de Software. Esta norma fornece uma estrutura para que os desenvolvedores de software possam “falar a mesma linguagem”, quando criam e gerenciam software. Os desenvolvedores podem usar a estrutura para adquirir, fornecer, desenvolver, operar e manter o software. É referência para contratação e fornecimento de serviços e produtos de software, facilitando o comércio internacional de bens e serviços em software, padronizando os processos entre as partes envolvidas.

O novo padrão estabelece uma arquitetura de alto nível para o Ciclo de Vida do Software, desde a definição de conceitos até o seu descarte. A arquitetura é construída através de processos e interfaces entre esses processos. Os processos são providos e

identificados com base nos princípios de modularidade e responsabilidade, em que cada processo é colocado sob a responsabilidade de uma parte ou participante dentro do ciclo de vida do software, independentemente da tecnologia utilizada. A modularidade permite que os processos sejam fortemente coesos, mas fracamente acoplados, ou seja, a quantidade de interfaces entre os processos é mínima. Cada processo deve ter uma estrutura interna suficientemente coesa e definida, para que possa ser executável. A parte que executa um processo tem a responsabilidade por todo o processo, mesmo que tarefas individuais possam ser realizadas por diferentes partes, facilitando a adaptação e a aplicação da norma em um projeto, no qual várias partes estejam formalmente envolvidas.

Devido à arquitetura flexível e independente de tecnologia, a norma é aplicável com qualquer modelo de Ciclo de Vida, método ou técnica de engenharia de software, ambientes de desenvolvimento ou linguagem de programação utilizada.

A norma está estruturada em três classes gerais: primária, suporte e organizacional. Na classe primária ou fundamental estão englobados os processos de aquisição, fornecimento, desenvolvimento, operação e manutenção, constituindo a alavanca primordial dentro do ciclo de vida. Dentro de suporte estão listados os processos de documentação, gerência de configuração, garantia da qualidade, verificação, validação, revisão, auditoria e resolução de problemas, servindo de apoio a outro processo, com um propósito distinto. Os quatro processos que compõem a classe organizacional são: gerenciamento, infra-estrutura, melhoria contínua e treinamento. Uma organização pode usar estes processos ao nível de cooperação, para estabelecer, implementar, gerenciar e melhorar seus processos de ciclo de vida, devendo ser adaptados ao contexto e características de cada projeto de software.

Cada processo é definido em termos de suas próprias atividades constituintes, e cada uma das atividades é subdividida em tarefas. O processo todo constitui-se de 74 atividades e 224 tarefas. Uma tarefa é expressa como um requisito, uma declaração própria, uma recomendação ou uma ação permissível. Um processo é particionado em atividades, empregando-se o princípio do “PDCA”, ou seja, PLAN-planejar as tarefas; DO-fazer as tarefas do plano; CHECK-checar, avaliar ou garantir essas tarefas; ACT-agir ou resolver problemas, executar ações corretivas.

A norma não pretende substituir o gerenciamento sistemático e disciplinado existente ou não nas organizações, nem as técnicas de engenharia e arquitetura de software. A norma meramente fornece uma estrutura, dentro da qual os processos, as atividades e as tarefas possam ser selecionados, planejados e executados, sendo baseada em um conjunto único de bloco de construção bem definido, que são os processos; o usuário deve, então, selecionar, adaptar e acoplar estes processos, contemplando um projeto em particular, para a realização de um propósito específico de seu negócio, recomendando fortemente que tal adaptação preserve a arquitetura, intenção e a integridade da norma.

3.3.2.1 Integração com outras normas ISO

No processo de aquisição e fornecimento de software, as atividades e tarefas envolvidas assemelham-se à norma ISO 9000-3. Dentro ainda do processo de aquisição pode-se verificar na fase de contratação, a integração com o processo de apoio com as atividades de revisão conjunta, auditoria, verificação e validação, até a conclusão deste processo.

No processo de desenvolvimento de software, a ISO 12207 integra-se com a norma ISO/IEC 9126, para avaliações da arquitetura do sistema e do software, bem como a integração do sistema.

No processo de manutenção, em que se definem as atividades do mantenedor, gerenciando as modificações no produto de software, para mantê-lo atualizado e operacional, incluindo a migração e a descontinuação do produto, pode-se utilizar da norma ISO/IEC 14764, em processo de tradução pela ABNT.

No processo de apoio, especificamente no processo de gerência de configuração, pode-se aplicar a norma ISO/IEC TR 15846, que trata de Gerência de Configuração. Para o processo de Garantia da Qualidade, que define as atividades para garantir objetivamente que os produtos e processos de software estejam em conformidade com seus requisitos especificados e adiram aos planos estabelecidos, vale-se da norma ISO 9001-3.

O processo de adaptação é um processo especial dentro dessa norma. Constitui-

se na base para a adaptação da norma para projetos de software, para um propósito específico de negócio. É composto de atividades como a identificação de ambientes do projeto, solicitação de entradas para o projeto, selecionar processos, atividades e tarefas bem como documentar decisões e as razões da adaptação, gerando processos institucionalizados, bem gerenciados.

3.3.2.2 Utilização da norma

A necessidade de definição de processos de software aderentes a padrões internacionais, facilitando o comércio de produtos e serviços, leva as organizações a buscarem ferramentas eficazes, que auxiliem na solução, porém surgem dificuldades, como: a definição do modelo adequado para preenchimento dessas necessidades, o desconhecimento de muitos aspectos da Engenharia de Software, inexistência de um processo de software que seja genericamente aplicado e, principalmente, a falta de processos disciplinados e gerenciados de ciclo de vida de software. Atualmente, as organizações buscam definir processos de software aderentes à norma ISO/IEC 12207, devido ao alto custo associado ao desenvolvimento de padrões particulares a cada organização e também porque a adoção de normas internacionais simplifica a relação entre clientes e fornecedores interessantes, no contexto de globalização da economia.

Alguns pré-requisitos devem ser observados para se fazer uso dessa norma:

a) disponibilidade da norma;
b) familiaridade com a norma;
c) familiaridade com as políticas organizacionais relevantes;
d) conhecimento geral de gerência de software, engenharia de software e modelos de ciclo de vida de software

Esta norma é considerada como um *framework* para processos de ciclo de vida com terminologia bem definida, isto é, diz o que tem que ter, mas não diz como.

Algumas limitações da norma:

a) não especifica detalhes de implementação;
b) não especifica detalhes de documentação;
c) não especifica modelo de ciclo de vida de desenvolvimento;
d) não especifica método de desenvolvimento de software;

e) não se aplica a “software de prateleira”;

f) não é certificadora.

3.3.2.3 Futuro da norma

A ISO 9000-3 estará passando para o SC7-Subcomitê 7, que irá mantê-la. A norma ISO/IEC 12207 será a referência, em relação aos processos para a norma ISO/IEC 15504 (projeto SPICE), que avalia os processos. A situação atual da norma encontra-se na fase de revisão, para ficar em conformidade com as referências de interação com outras normas.

Mudanças estão em discussão no âmbito da SC10, subcomitê de software da ABNT, CE-21, Comissão de Estudos 21:

a) inclusão de processos:
   - de reuso
   - de medição
   - de gerência de riscos
   - de avaliação de qualidade de produto
   - gerência de bens
   - usabilidade

b) trabalhos em relação a processos de:
   - ISO/IEC 12207 - visão geral - abrangência;
   - ISO/IEC 14764 – processo de manutenção;
   - TR 15846 – gerência de configuração;
   - TR xxxxx – gerência de projeto;
   - ISO/IEC 15288 – processos de ciclo de vida de SISTEMA.

3.3.3 CMM - CAPABILITY MATURITY MODEL

Este modelo foi desenvolvido pela Software Engineering Institute-SEI da Carnegie Mellon University, atendendo a clientes como o Department of Defense-DOD dos EUA. A divulgação dos trabalhos começou em 1991, embora o começo dos estudos partisse de 1986. O governo americano, em 1984, encomendou uma revisão, não publicada, dos 17 maiores contratos de desenvolvimento de software do DOD, onde encontraram prazos estipulados em média de 28 meses, ultrapassados em 20 meses; um

projeto de quatro anos não havia sido entregue em sete anos, constatando-se, também, que nenhum projeto estava no prazo estabelecido. A partir desse cenário, criou-se a necessidade de investir em um padrão de qualidade na área de desenvolvimento de software, fundando-se o SEI, um instituto de pesquisa e desenvolvimento com a missão de liderar iniciativas para melhorar a qualidade do desenvolvimento de softwares. (CARAM, 2001).

Em 1987, o SEI lançou uma descrição de ambiente de maturidade de processo de software e desenvolveu dois métodos: sendo um, a avaliação do processo de software, com o objetivo de determinar o nível do processo atual de desenvolvimento de software de uma organização; e o outro, a avaliação da capacidade de software, com o objetivo de identificar fornecedores qualificados para esse desenvolvimento, e, também, um questionário de maturidade para avaliar a maturidade do processo de software. Após quatro anos de experiência, esses ambientes de maturidade evoluíram ao hoje denominado CMM-Capability Maturity Model, um modelo de maturidade de processos.

O CMM tem seu foco no processo de software, por entender que a qualidade de um sistema de software é fortemente influenciada pela qualidade do processo utilizado para desenvolvê-lo e mantê-lo. Da mesma forma, a capacidade do processo de software descreve o conjunto de resultados esperados, que pode ser atingido quando se segue o processo de software estabelecido. Já, a maturidade do processo de software é o quanto um processo específico é explicitamente definido, gerenciado, medido, controlado e efetivo, implicando um potencial de crescimento da capacidade e indica tanto a riqueza do processo de software de uma organização quanto a consistência na qual o processo é aplicado em todos os seus projetos (PAULK, 1993). O modelo CMM, basicamente, propõe a avaliação da capacidade e maturidade de uma organização e indica diretrizes para a melhoria dos processos.

As versões atualmente em desenvolvimento, ainda não divulgadas, enfocam o produto de software. Inclui um conjunto de ferramentas de diagnóstico e avaliação do produto, avaliação de software, evolução da maturidade do software, riscos de uma organização, avaliação do processo de capabilidade e evolução do processo de software e também treinamento.

No modelo CMM as organizações de software são enquadradas em 5 níveis de maturidade definidos, baseando-se no princípio de que a melhoria contínua do processo

se dá em pequenos passos evolutivos, e não em inovações revolucionárias. Cada nível apresenta as características da organização e as áreas-chave de processos necessários para que seja classificada em um determinado nível, alcançando um determinado objetivo. Esses níveis de maturidade definem uma escala para medir a maturidade do processo de software e também para avaliar a capacidade deste, significando o estabelecimento de diferentes componentes, que resultam num crescimento da capacidade do processo de uma organização. Esta estrutura, em níveis do modelo, está baseada nos princípios de qualidade propostos por Shewart, Deming, Juran e Crosby (PAULK, 1993).

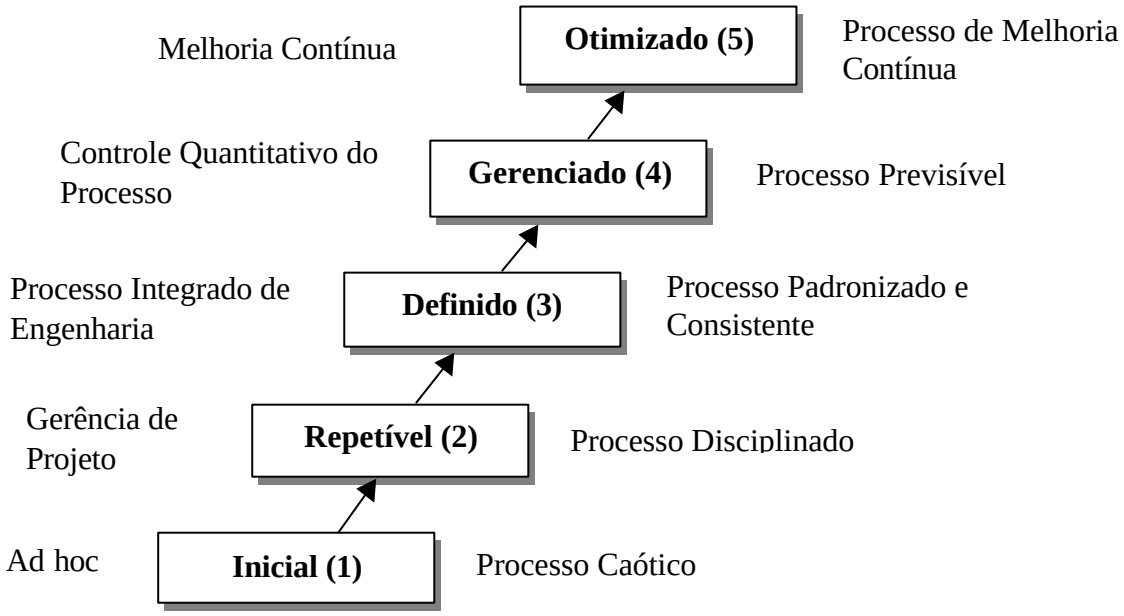

FIGURA 01 – Visão Geral dos Níveis de Maturidade do CMM (PAULK, 1993).

O modelo CMM e a série ISO 9000 mostram conceitos comuns como a qualidade e a gerência de processos. Os dois apresentam conceitos similares e intuitivamente correlatos. O CMM descreve os princípios e práticas fundamentais em gerência de processos de software e serve para empresas que queiram aperfeiçoar os seus processos para melhorar ou procurar um caminho, aperfeiçoar um processo caótico ou disciplinar um processo. O nível de maturidade no qual o modelo atua são os níveis de evolução para alcançar o nível de excelência em processos. Cada nível de maturidade provê uma camada, para a melhoria contínua do processo de software.

Exceto para o nível 1, cada nível de maturidade é decomposto em muitas áreas-

chave de processo, que indicam as áreas em que serão focadas as melhorias no processo. Cada área-chave de processo-KPA identifica um grupo de atividades relacionadas que, quando processadas coletivamente, ativam um conjunto de metas consideradas importantes, para incrementar a capacidade do processo. Cada KPA é organizado pelas características comuns que são atribuídas para indicar se a implementação e a institucionalização efetiva, repetidamente, é permanente e sistematizada. As características comuns incluem práticas sobre política e gerenciamento de uma organização, as condições para a implementação do processo de software competente, planos e procedimentos para ações corretivas, métricas de qualidade de processo e produto de software, as ferramentas e recursos organizacionais e a garantia de que as atividades executadas estão em concordância com os planejados, com o estabelecimento da gerência das revisões e auditorias.

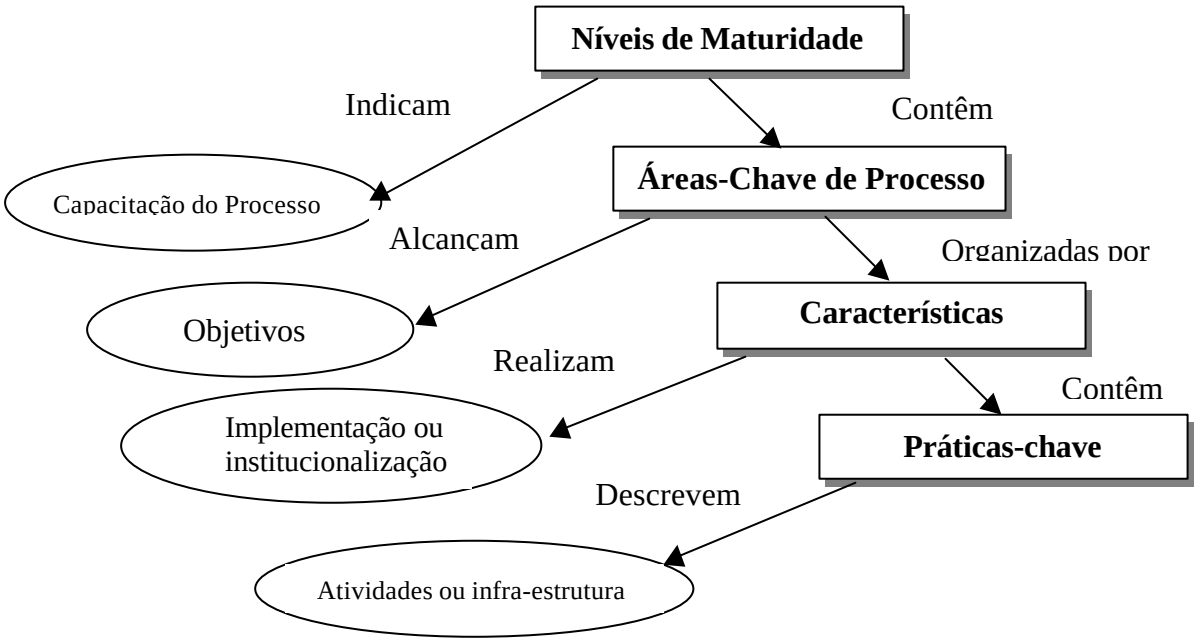

FIGURA 02 – Estrutura do CMM (MACHADO, 2001).

O nível 1 é caracterizado como um nível *ad hoc* ou até mesmo como caótico. A maioria das organizações de software enquadram-se neste nível, caracterizadas como desenvolvimento sem um padrão definido de processos e nem gerenciamento, contando apenas com o conhecimento claro de entradas e saídas. A qualidade do produto final depende de esforços individuais, em que se valorizam a criatividade e a arte, sem qualquer nível de controle ou gerenciamento definidos e documentados explicitamente.

Este nível não possui as áreas-chave de processo, por ser o nível inicial em que existem poucos processos estáveis ou que estejam em uso.

No nível 2, chamado de nível repetível, cada KPA enfoca o projeto do software e o controle e gerenciamento do projeto base estabelecido. Quanto a processos, existem estimativas e planejamentos estáveis e documentados, podendo repetir tarefas em um processo gerenciado e disciplinado, com os problemas sendo percebidos e corrigidos, caracterizando a melhoria contínua. A busca da qualidade do produto final depende das pessoas com o suporte do sistema de gerenciamento, de treinamentos e cooperação entre as partes envolvidas. O objetivo é disciplinar e melhorar os processos básicos de gerência de projeto, para planejar e acompanhar custos, cronogramas e funcionalidades, estabelecendo padrões que possam ser repetíveis. As áreas-chave deste nível são: gerenciamento de requisitos, planejamento de projeto de software, supervisão e acompanhamento do projeto de software, gerenciamento de subcontratos de software, garantia de qualidade de software e gerenciamento de configuração de software.

O nível 3, considerado o nível definido, endereça ambos, o projeto e as saídas organizacionais, como as diretrizes organizacionais estabelecidas e infra-estrutura que institucionaliza a engenharia de software efetiva e o gerenciamento do processo, por meio de todos os projetos. Os problemas são antecipados e prevenidos, ou seu impacto é minimizado, o entendimento do processo de software é satisfatório, sendo documentadas adequadamente todas as atividades de desenvolvimento. As pessoas são integradas em equipes de projeto e o treinamento é planejado e realizado de acordo com os perfis de cada profissional, disseminando e consolidando o processo-padrão estabelecido. Este nível é constituído das seguintes áreas-chave de processo: foco do processo da organização, definição do processo da organização, programa de treinamento, gerenciamento integrado de software, coordenação entre grupos e ponto de revisão.

No nível 4, o nível gerenciado, as KPAs estabelecem quantificações e qualificações em ambos, o processo de software e os produtos de software sendo construídos. Estabelece-se uma política de medição, adotando-se métricas consistentes, para que seja possível compreender quantitativamente e avaliar de forma segura a qualidade do processo. Os KPAs deste nível são: gerenciamento quantitativo do processo e gerenciamento qualitativo de software.

O nível 5, chamado de nível otimizado, cobre as saídas que ambas, a organização e os projetos devem endereçar para a busca contínua da qualidade. Aborda a melhoria contínua, identificando as causas do defeito e a prevenção da ocorrência, o implemento das medições dos processos para a melhoria da qualidade do software, aumentando a produtividade e o implemento de novas tecnologias, identificando os benefícios e transferindo-os para a organização para a melhoria contínua e sistemática.

Claramente, há uma forte correlação entre ISO 9001 e CMM, embora algumas normas na ISO 9001 não sejam abertas na CMM e algumas normas na CMM não tenham correspondente na ISO 9001. A diferença básica é a ênfase do CMM sobre aperfeiçoamento contínuo do processo e a ISO 9001 trata sob critérios mínimos para um aceitável sistema da qualidade, sob a ótica de alguns especialistas, e para outros, há a sustentação de que, com auditorias freqüentes e regulares, pode-se manter o entendimento da ISO como melhoria contínua do processo (PAULK, 1993).

Conforme a comunidade internacional, uma organização que está sob a certificação ISO 9001, ela certamente estará no nível 2 do CMM, porém, esta observação enfatiza a necessidade de visão da experiência e conhecimento da auditoria. O que pode ocorrer, também, é uma organização no nível 3 do CMM não ter certificação ISO, na visão e experiência de auditorias, pelas características de cada modelo, mas pode ter poucas dificuldades em obter certificação ISO 9001.

Em qualquer caso, a construção de uma vantagem competitiva será focada na melhoria contínua de processos e produtos e não sobre a obtenção de um escore, ou se o escore está em um nível de maturidade ou dentro de uma certificação. A organização deve desenvolver seus sistemas de qualidade sempre focada em seus negócios, em sua atuação geográfica, em conformidade com um contexto maior de mercado.

3.3.4 SPICE OU ISO/IEC 15504

SOFTWARE PROCESS IMPROVEMENT AND CAPABILITY DETERMINATION-SPICE foi criado com o objetivo de lançar um padrão internacional para avaliação e melhoria de processo de software, também conhecido como modelo SPICE, unificando propostas existentes.

SPICE nasceu dos estudos realizados com o aprovação do Comitê de Engenharia de Software da ISO, sobre a necessidade de padrões globais de melhoria de

processo de desenvolvimento de software e padrões de avaliação destes. Reuniu-se em centros regionais significativos de desenvolvimento: o Pacific Rim (Austrália), Europe (Inglaterra), Canadá e EUA. Atualmente publicada como um relatório técnico (TR) da ISO/IEC, com previsão de publicação como norma em 2002, formando um conjunto de nove documentos, referenciados como ISO/IEC TR 15504-1 a ISO/IEC TR 15504-9, sendo combinada para a geração de um conjunto de apenas cinco documentos.

O conjunto de nove documentos são:

a) 15504-1 TR2 Part 1: conceitos e guia de introdução (informativo);
b) 15504-2 TR2 Part 2: um modelo de referência para processos e capacidade de processos (normativo);
c) 15504-3 TR2 Part 3: executando uma avaliação (normativo);
d) 15504-4 TR2 Part 4: guia para execução de uma avaliação (informativo);
e) 15504-5 DTR Part 5: um modelo de avaliação e guia de indicadores (informativo);
f) 15504-6 TR2 Part 6: guia para competência de assessores (informativo);
g) 15504-7 TR2 Part 7: guia de uso para melhoria de processos (informativo);
h) 15504-8 TR2 Part 8: guia de uso para determinar a capacidade de processo de fornecedor;
i) 15504-9 TR2 Part 9: vocabulário (informativo).

A 15504 define um *framework* para a avaliação de processo de software, baseado nas melhores práticas de vários modelos de processos existentes, tais como: CMM, Trillium, e Bootstrap, bem como nas normas ISO 9001/9000-3 e ISO/IEC 12207, podendo estabelecer um comparativo entre si, definindo as características de compatibilidade. A 15504 caracteriza-se como um modelo de referência, com um conjunto universal de processos fundamentais para a engenharia de software e um roteiro racional para avaliação e melhoria de cada processo ou capacidade de processo, podendo cada modelo detalhá-los, determinando os níveis de capacidade.

O levantamento das necessidades, coordenado pelos centros regionais, foi baseado no estabelecimento de arquiteturas deste projeto em duas dimensões: cada dimensão representa uma perspectiva diferente do processo de gerenciamento do software. Uma dimensão consiste em práticas básicas. As práticas básicas são definidas como engenharia de software ou o gerenciamento das atividades que endereçam os

propósitos de processos particulares. Práticas básicas são agrupadas em processos, os quais são agrupados em categorias de processos. Um exemplo de processo seria o Requisito de Desenvolvimento de Sistemas e Projeto. A prática básica a que pertence este processo inclui: requisitos de especificação do sistema, descrição da arquitetura do sistema e determinação da estratégia de atualização.

A outra dimensão consiste em práticas genéricas. Práticas genéricas são uma implementação ou práticas institucionalizadas, que melhoram a capacidade de executar os processos. Estas práticas são agrupadas em *Common Features*, as quais são agrupadas novamente em *Capability Levels.* Um exemplo de *Common Features s*ão execuções-padrão. A prática genérica a que pertence esta *Common Feature* estipula que dados para melhorar o processo devem ser registrados. Bases e práticas genéricas podem ser fixadas durante a avaliação.

O SPICE pode ser utilizado por organizações de software envolvidas em planejar, monitorar, gerenciar, controlar e melhorar os processos de aquisição, fornecimento, desenvolvimento, operação, evolução e suporte de software

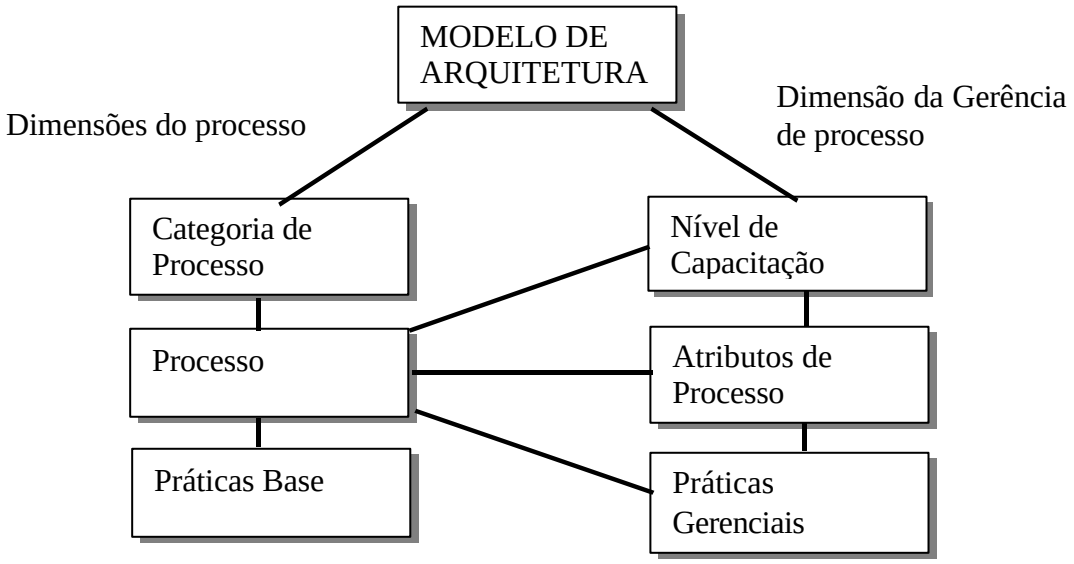

FIGURA 03 – Arquitetura do modelo SPICE (Futura Norma ISO 15504).(ESI, 2000).

O modelo de referência SPICE, baseado na ISO/IEC 12207 é distribuído em três blocos comuns: os processos fundamentais ou básicos, com as atividades de aquisição, fornecimento, definição de requisitos, desenvolvimento, operação e manutenção de software e do sistema; os processos de apoio, com as atividades de

documentação, gerência de configuração, garantia da qualidade, verificação, validação, revisão conjunta, auditoria e resolução de problemas; e os processos organizacionais, que compreendem as atividades de gerência, gerência de projeto, gerência de qualidade, gerência de risco, alinhamento organizacional, gerência de recursos humanos, infra-estrutura, reuso de soluções e melhoria contínua de processos.

O modelo descreve atributos de processo agrupados em níveis de capacitação, podendo ser aplicados a qualquer processo.

O nível zero, nível não executado ou incompleto, descreve um ambiente de falha geral de execução das práticas básicas no processo. Não há uma identificação fácil dos produtos e serviços ou das saídas do processo.

O nível 1, ou o nível executado informalmente, descreve que as práticas básicas do processo são geralmente executadas, mas não são rigorosamente planejadas e traçadas. Execução depende de conhecimentos e esforços individuais. Há produtos e serviços identificáveis no processo.

No nível 2, ou o nível planejado e traçado, as execuções das práticas básicas no processo são planejadas e traçadas. Execuções de acordo com os procedimentos especificados são verificados. Produtos e serviços estão conforme o padrão especificado e requerido.

No nível 3, chamado também de nível bem definido ou estabelecido, as práticas básicas são executadas de acordo com o uso de processos bem definidos, aprovados, testados e documentados.

O nível 4, o nível de controle de quantidade ou previsível, descreve as medidas detalhadas de execuções, que são coletadas e analisadas, seguindo uma quantidade variável de capacidades de processo e uma improvável habilidade para predizer resultados. Performances: são objetivamente gerenciadas. A qualidade do produto e dos serviços é quantitativamente conhecida.

No nível 5, caracterizado como de melhoria continua ou otimizada, os processos efetivos e metas eficientes para execuções são estabelecidas, baseadas em metas de negócios da organização. Processos contínuos de melhoria garantem que as metas serão atingidas pelo *feedback* estabelecido.

Nos trabalhos atualmente em andamento no Projeto SPICE, todo o enfoque está voltado para gerar um documento com o resultado do que está sendo

industrialmente usado, estando particularmente atento a:

a) critérios de evolução para o estabelecimento de modelos de processo conformes e os métodos de avaliação dentro do SPICE (isto é, diferentes métodos de avaliação de conformidades e modelos serão usados).
b) forma como o processo SPICE e práticas genéricas serão agrupadas, e se serão expressos em uma ordem apropriada;
c) evolução da extensão a qual diferentes equipes estão de fato fazendo julgamentos equivalentes, e
d) evolução do esforço requerido para avaliar a base SPICE.

A próxima fase será uma tentativa de expandir a participação igualmente extensa, e será concentrada na reunião de evidências de valor comercial da melhoria de processo de software, usando o método SPICE.

3.3.4.1 Integração das normas e modelos de maturidade

O uso combinado do modelo de maturidade ISO/IEC TR 15504, do Capability Maturity Model-CMM e da norma ISO/IEC 12207, gerando um processo padrão, pode ser realizado com o mapeamento dos atributos dos modelos de maturidade em cima da arquitetura da ISO 12207, definindo para os modelos: processos, atividades e tarefas, usando o modelo ISO 12207, os processos, as práticas base e práticas gerenciais, usando ISO 15504 e, no modelo CMM, as KPAs e atividades organizacionais. (ROCHA & MACHADO, 2000).

Algumas observações sobre o uso combinado das três ferramentas, conforme Rocha e Machado (2000):

a) o foco do CMM é o desenvolvimento de software;
b) nível de processo (SPICE) versus nível organização (CMM);
c) maior rigor imposto pelo CMM;
d) o processo de treinamento possui enfoques diferentes;
e) algumas atividades, como análise dos requisitos do sistema, projeto da base de dados, instalação de software e sistemas de garantia de qualidade não são considerados pelo CMM ;
f) algumas atividades, como projeto da base de dados, avaliação da configuração, migração e documentar versões, não são tratadas na ISO TR

15504;

g) uma KPA (CMM) pode conter atividades presentes em mais de um processo da ISO/IEC 12207;
h) algumas KPAs não apresentam uma correlação bem definida com os processos definidos pela ISO/IEC 12207;
i) os processos básicos e estendidos (15504) foram correlacionados com os processos da 12207 e os componentes em atividades.

Os modelos de maturidade e a norma ISO/IEC 12207 definem atividades que deverão ser incorporadas a um processo de software, observando os níveis de capacitação e os guias de implementação. Um mapeamento cruzado entre os modelos de maturidade e a norma ISO 12207 permite que o engenheiro de software analise cada atividade a ser definida, fazendo comparação e incorporando-os a processos definidos, particulares de cada organização.

3.3.5 ISO/IEC 9126

Norma usada para avaliação de produto de software, sendo um guia de características de qualidade e diretrizes para o seu uso, na evolução de produtos de softwares. Sendo um *framework*, não fornece as subcaracterísticas e nem as métricas, nem métodos para medição, pontuação ou julgamento. As características da norma podem ser aplicadas em qualquer tipo de software, incluindo os programas e dados contidos em *firmware* de computador, podendo ser aplicadas por quem trabalha com aquisição, desenvolvimento, uso, suporte, manutenção e auditoria de software.

Os aspectos técnicos para avaliação da qualidade do produto de software são abordados em três normas:

a) ISO/IEC 9126 - Características de qualidade de software;
b) ISO/IEC 14598 - Guias para avaliação de produto de software, e
c) ISO/IEC 12119 - requisitos de qualidade e testes de pacotes de software.

A norma ISO/IEC 9126, de 1991, definia um modelo de qualidade e um processo resumido de avaliação de produto de software. O segundo ciclo desenvolvido tem a seguinte estrutura de documentos:

a) 9126-1 – modelo de qualidade de software;

b) 9126-2 – descrição das métricas externas;

c) 9126-3 – descrição das métricas internas.

O conjunto de seis características básicas de qualidade com algumas subcaracterísticas que um software deve ter:

a) funcionalidade: mede as capacidades do software, evidenciando o conjunto de funções que atendem às necessidades explícitas e implícitas para o fim a que se destina o produto de software. São avaliadas características como padrões funcionais, adequabilidade, precisão, conformidade, segurança de acesso e interoperabilidade;

b) confiabilidade: característica que determina o desempenho do produto, verificado ao longo do tempo, dentro de condições estabelecidas. São observados fatores como segurança, ausência de falhas e resultados corretos, bem como subcaracterísticas como: recuperabilidade, tolerância ao mau uso por parte do usuário e maturidade;

c) usabilidade: também se usa o termo utilizabilidade. Avalia o esforço necessário ao uso do software e ao seu aprendizado, evidenciando a facilidade de uso do produto de software. Definem-se interfaces com o usuário, documentação clara e material de treinamento adequado;

d) eficiência: também considerada como performance ou desempenho. Aborda o esforço para estabelecer compatibilidade entre os recursos e os tempos envolvidos para um uso eficaz. Consideram-se, neste item, as subcaracterísticas como: tempo de resposta, número de usuários atendidos simultaneamente e quantidade de recursos do sistema alocados;

e) manutenibilidade: define as características através dais quais se mede o esforço despendido para fazer alterações específicas, atualizações ou correções no software. O esforço mede-se pela análise de facilidade de manutenção, rapidez, custo, tempo médio entre falhas (MTBF), tempo médio de reparo (MTTR);

f) portabilidade: avalia-se o esforço para transferência de ambientes proprietários, evidenciando a possibilidade de se utilizar o produto em

diversas plataformas ou sistemas operacionais.

A importância de cada característica descrita depende da classe do software, por exemplo, confiabilidade é mais importante para um sistema de missão crítica, a eficiência é mais importante para o software de tempo real, a usabilidade é mais importante para um software interativo em relação ao usuário final (NBR 13596).

O Capítulo 5 desta norma trata das diretrizes para o uso das características de qualidade de software e realização da avaliação de seus processos. O processo é constituído do estágio de utilização em que se definem os requisitos de qualidade na avaliação de produtos de software, estabelecendo as visões de cada parte, dentro de um modelo padronizado e sistematizado.

Na avaliação incluem-se a medição, pontuação e julgamento e também a definição de requisitos de qualidade, avaliação de especificação de software, para verificar os requisitos de qualidade durante o desenvolvimento, descrição de particularidade e atributos do software implementado, avaliação de software desenvolvido antes da entrega, e a avaliação antes da aceitação, podendo ser aplicado a cada fase do ciclo de vida de cada componente do produto de software.

O estágio de visão da qualidade de software, inclui a visão do usuário, da equipe de desenvolvimento e a visão do gerente. Na visão do usuário, o software é avaliado sem o conhecimento dos aspectos internos ou de como foi desenvolvido, no seu desempenho e nos efeitos do uso como a confiabilidade, a eficácia, a usabilidade e a portabilidade, por exemplo. Na visão da equipe de desenvolvimento, é avaliado baseado em métricas diferentes, para cada ciclo de vida do software, dentro das mesmas características da visão do usuário ou de quem faz a manutenção do software. Na visão do gerente, as características são medidas conforme os requisitos comerciais, conforme as características do negócio, observando os limites de custo, recursos humanos, prazos e recursos organizacionais.

Na definição do modelo de avaliação do processo, não são mostrados procedimentos detalhados, como a análise e validação de métricas e, sim, as regras e práticas gerais para o estabelecimento de procedimentos disciplinados. O processo é constituído de três estágios: definição de requisitos de qualidade, preparação da avaliação e procedimento de avaliação.

A nova estrutura desta norma, em elaboração na ISO/IEC, evidencia a definição de métricas e trata-as como o ponto nevrálgico da qualidade de software. A nova estrutura divide-se em quatro partes: definições das características, as métricas externas e internas para medição das características definidas e a qualidade em uso do software que representa a qualidade segundo o enfoque dos resultados de uso do software em um determinado contexto de uso.

As características de qualidade em uso são a efetividade ou eficácia, a produtividade, a segurança e a satisfação. Pode-se adotar a denominação de usabilidade ou ergonomia de software. A avaliação de qualidade em uso ocorre após a liberação do software, objetivando revisar seu uso ou simular a situação em projetos semelhantes, com usuários reais, em um ambiente real, com dados reais. Pode-se avaliar o software, utilizando-o como parte de um sistema, num ambiente simulado, com dados de teste e operadores envolvidos na simulação, identificando necessidades que não foram explicitadas na definição dos requisitos do sistema. Dentre os métodos de medição, utiliza-se o *feedback* dos usuários, observação do comportamento do usuário ou também medições locais.

3.3.6 ISO/IEC 14598

Esta norma provê um conjunto de guias que orientam o planejamento e a execução de um processo de avaliação da qualidade do produto de software. A série 14598 é composta de:

a) 14598-1: Visão geral da estrutura dessa série de normas e dos processos de avaliação;
b) 14598-2: Planejamento e gerenciamento do processo de avaliação;
c) 14598-3: Processo para desenvolvedores, enfoque no uso de indicadores para prever qualidade final;
d) 14598-4: Processo para adquirentes na avaliação no processo de seleção de pacotes ou produtos de software, abordando o processo de apoio de aquisição, segundo a norma ISO/IEC 12207;
e) 14598-5: Processo para avaliadores, descreve o ciclo de vida de avaliação, com definição das atividades, inclusive a relação entre avaliador e

requisitante, objetivando a produção de relatório de avaliação;

f) 14598-6: Documentação de módulos de avaliação, que são pacotes estruturados de métodos e ferramentas para o apoio às partes relacionadas.

Há um forte relacionamento entre as normas ISO 9126 e a ISO 14598, no contexto de suas partes e seus guias. A norma 9126 fornece um modelo de qualidade e as métricas interna, externa e qualidade em uso, e a norma 14598 trata dos processos de avaliação dentro do modelo de qualidade descrito na 9126.

A parte 2 da norma, que trata de planejamento e gestão, organiza as atividades de suporte à avaliação, fornecendo requisitos e recomendações e orientações para o exercício da atividade. O público-alvo são a gerência de tecnologias de avaliação, equipe de suporte à avaliação de software, a gerência de desenvolvimento de software e a equipe de garantia de qualidade. A norma orienta para a existência de função de suporte à avaliação, que trabalha na obtenção e elaboração de padrões e ferramentas de apoio, na avaliação da eficácia da aquisição e desenvolvimento de software e facilitação de transferência de tecnologia, podendo ser aplicada tanto interna quanto externamente à organização.

A parte 6, que trata da documentação de módulos de avaliação, é um conjunto estruturado de dados e instruções de uso para avaliação, possibilitando a repetibilidade, reprodutibilidade e imparcialidade da avaliação. Basicamente, especifica o método, o procedimento e o formato de relatório de avaliação de uma determinada característica da qualidade, encapsulando-os de forma padronizada e sistematizada, facilitando a obtenção de informações.

A parte 5 trata da avaliação de produto de software, definindo os processos para avaliadores, objetivando principalmente o fornecimento de resultados quantitativos sobre a qualidade de produto de software, que sejam compreensíveis, aceitáveis e confiáveis por quaisquer das partes envolvidas. O processo de avaliação é descrito como um procedimento passo-a-passo, de tal forma que os requisitos possam ser expressos em termos de características de qualidade, como definido na NBR 13596/ISO 9126. Por ser um processo abstrato e genérico, pode ser aplicado a todos os processos fundamentais do ciclo de vida definidos na ISO/IEC 12207, em que alguns processos de apoio de ciclo de vida estão relacionados diretamente ao processo de avaliação, como o processo de garantia da qualidade, verificação, validação, revisão conjunta e auditoria. Pode

também ser utilizado para verificar conformidade com a norma ISO/IEC 12119. Essa parte da norma é composta de itens, como as partes envolvidas, o processo da avaliação descrevendo as entradas e saídas, a avaliação no ciclo de vida, definição dos requisitos em que se trata das responsabilidades das partes (ABNT, 1999).

O capítulo 6 desta norma descreve os requisitos de processo da avaliação, incluindo as responsabilidades das partes, elaboração dos requisitos com o relatório da aprovação ou não do conteúdo, define a abrangência da avaliação e as medições a serem realizadas, estabelece um plano de avaliação, revisão conjunta e destinação dos dados e documentos de avaliação.

Nos anexos dessa norma, constam o modelo de relatório de avaliação, que é normativo, os níveis de avaliação com sugestão de técnica de seleção, definição dos componentes de produto de software, as relações de interação entre requisitante e avaliador e os termos de contrato de avaliação (ABNT, 1999).

3.3.6.1 O futuro da norma

Está em estudo, no âmbito da ISSO, a elaboração de um novo ciclo de evolução, baseado nas normas que tratam da avaliação de processos e produtos de software, as normas ISO/IEC 9126/NBR 13596 e ISO/IEC 14598. A nova arquitetura foi denominada de SQUARE-Software Product Quality Requirements and Evaluation, recebendo a numeração a partir de 25000 (SCALET, 2001).

A divisão de qualidade de produto é composta de duas ferramentas, com numeração 25000 e 25001. A 25000 trata de um modelo de referência e um guia de uso do SQUARE, a 25001 é um guia de planejamento e gerência da qualidade.

A divisão do modelo de qualidade contém um guia com a definição do modelo de qualidade, com a numeração 25010.

A divisão de métricas de qualidade é composta dos guias: 25020, que trata dos modelos de referência de métricas e guias; 25021, define as métricas básicas; 25022, descreve as métricas internas; 25023, descreve as métricas externas; 25024, as métricas de qualidade em uso; e a 25025, módulo de documentação da evolução.

Compõe o conjunto uma divisão de requisitos de qualidade, com a definição dos requisitos com a numeração 25030.

A divisão de evolução da qualidade é composta de um guia genérico de

processo de avaliação, com a numeração 25040; a 25041, um guia de processo do desenvolvedor; 25042, um guia de processo do adquirente; e 25043, um guia de processo do avaliador.

Não se pretende substituir ou extinguir as normas existentes, mas investir em estudos na reformulação das normas, visando à evolução dos conceitos, com conseqüente qualificação dos produtos de software.

3.3.7 ISO/IEC 12119

Essa norma estabelece requisitos de qualidade para pacotes de softwares, além de instruir como testar um software a partir dos requisitos estabelecidos. O escopo refere-se ao software na forma final, oferecido no mercado e, não, ao processo de produção do software. São exemplos de pacotes de software: processadores de texto, planilhas eletrônicas, bancos de dados, softwares gráficos, programas para funções técnicas ou científicas e programas utilitários (ABNT, 2000).

Incluem-se como possíveis usuários desta norma:

a) fornecedores que estejam:
   - especificando os requisitos para um pacote de software;
   - projetando um modelo para descrever produtos;
   - julgando seus próprios produtos;
   - emitindo declarações de conformidade (guia 22);
   - submetendo produtos à certificação ou à obtenção de marcas de conformidade (guia 23);
b) laboratórios de teste que terão de seguir as instruções durante a execução de testes para certificação ou para emissão de marca de conformidade (guia 25);
c) entidade de credenciamento, que credencia entidades de certificação e laboratórios de teste (guias 40 e 58);
d) auditorias, quando julgam a competência de laboratórios de teste (guia 58);
e) compradores que pretendam:
   - comparar seus próprios requisitos com os descritos na norma;
   - comparar os requisitos necessários para executar uma determinada tarefa com a informação presente nas descrições de produtos existentes;

- procurar por produtos certificados;
- verificar se os requisitos foram atendidos;

f) usuários que pretendam se beneficiar com produtos melhores.

A norma trata de todos os componentes dos produtos disponíveis aos usuários tais como: documentação, manual de instruções e guia de instalação. Estabelece um conjunto de seis características de qualidade de produto de software: funcionalidade, confiabilidade, usabilidade, eficiência, portabilidade e manutenibilidade.

A norma está estruturada em duas partes: os requisitos de qualidade e as instruções de teste.

3.3.7.1 Requisitos de qualidade

São definidos os requisitos necessários para que um pacote de software tenha qualidade. São definidas a necessidade de que cada pacote de software tenha uma descrição do produto e documentação do usuário, requisitos para a descrição do produto, requisitos para a documentação do usuário, requisitos para programas e dados.

A descrição do produto é um documento único, que deve estar disponível para pessoas interessadas, independente de sua aquisição, com o objetivo de auxiliar o usuário ou comprador na avaliação da adequação do produto para seus fins, informações úteis para venda e base para testes. Deve ser compreensível, completa, livre de inconsistências internas; devem ser testáveis e corretas, conter uma visão geral do uso adequado, especificar as identificações e indicações do produto, fornecedor, tarefas, requisitos de hardware e software, interface com outros produtos, instalação, suporte, manutenção e os itens entregues. Deve ainda conter declarações sobre funcionalidade, confiabilidade e usabilidade e opcionalmente, sobre dados de eficiência, manutenibilidade e portabilidade.

Na documentação do usuário os requisitos são:

a) completitude: todas as funções estabelecidas na descrição do produto e todas as funções do programa a que os usuários tenham acesso devem ser completamente descritas;
b) correção: todas as informações devem ser corretas e convém que sua apresentação não contenha ambigüidades nem erros;
c) consistência: não devem apresentar contradições internas e nem com a

descrição do produto;

d) inteligibilidade: convém que seja inteligível pela classe de usuários que normalmente executa a tarefa, utilizando uma seleção apropriada de termos, gráficos, explicações detalhadas e citando fontes úteis de informações;
e) apresentação e organização: convém que possua boa apresentação e organização, de tal modo que quaisquer relacionamentos sejam facilmente identificáveis. Convém que tenha um índice analítico e remissivo.

Nos requisitos de qualidade de programas e dados são descritas características como:

a) funcionalidade: com o manual de instalação fácil e verificável, presença de funções executáveis conforme a documentação, correção nas execuções e ausência de contradições internas e nem com a documentação;
b) confiabilidade: o sistema não deve ficar fora do controle do usuário e nem corromper ou perder os dados, mesmo em condições limite, excluindo as falhas decorrentes de falha de hardware ou de sistema operacional;
c) usabilidade: recomenda-se investigar a possibilidade de aplicar as edições mais recentes da série ISO 9241, em particular às partes 10 e 13. São descritos os requisitos de inteligibilidade, apresentação, organização e operacionalidade;
d) eficiência: é uma característica opcional, não sendo exigida, mas deve estar em conformidade com as citadas na descrição do produto;
e) manutenibilidade: é uma característica opcional, se usada, deve estar em conformidade com a descrição do produto;
f) portabilidade: também é opcional, devendo estar em conformidade com a citada na descrição do produto.

3.3.7.2 Instruções para teste

São recomendações de como um produto deve ser testado em relação aos requisitos de qualidade definidos na norma. Incluindo os testes das propriedades necessárias a todos os produtos de mesmo tipo, os testes de propriedades especificadas na descrição do produto, o teste de inspeção de documentos e o teste funcional (teste caixa-preta). O teste funcional não está incluído, porque requer a disponibilidade de

código-fonte, tampouco a avaliação ergonômica do ambiente computacional, por não se enquadrar no escopo da norma (ABNT, 1999).

O capítulo referente às instruções para teste é composto de:

a) pré-requisitos de teste: devem estar presentes todos os itens dos produtos identificados na descrição do produto, os componentes do sistema e o treinamento, se mencionado na descrição do produto;
b) atividades de teste: verificar a conformidade com os descritos em requisitos de qualidade do produto, da documentação do usuário e de programas e dados;
c) registros de teste: os testes devem ser registrados para que permitam a repetição (ABNT ISO/IEC guia 25), devendo incluir um plano de teste ou especificação de teste, contendo os guias de teste, os resultados associados com os guias de teste e a identificação da equipe de teste;
d) relatório de teste: os objetos e resultados de teste devem ser resumidos em um relatório de teste seguindo uma estrutura definida conforme a norma;
e) teste de acompanhamento: os testes das partes modificadas devem ser novamente repetidos, como se fosse um produto novo e as partes inalteradas, mas com possibilidade de serem influenciadas, também devem ser testadas, incluindo um teste completo considerando-se casos de teste com seleção de amostragem.

3.3.8 TRILLIUM

O TRILLIUM é um modelo de avaliação de organizações desenvolvido pela Bell Canadá.

A procura de produtos de software ocasiona riscos que precisam ser avaliados e gerenciados. Este método especifica o produto de software procurado, definições de fatores de risco, técnicas de segurança, fatores de risco na avaliação e consideração da empresa.

O modelo TRILLIUM cobre todos os aspectos do desenvolvimento de software em seu ciclo de vida, mais sistemas e produtos desenvolvidos e atividades de suporte, e um significativo número de atividades de negócios relatados.

Embora o TRILLIUM tenha sido designado para ser aplicado em sistemas de

softwares pacotes, tal como de sistemas de telecomunicações, muito do modelo pode ser aplicado em outros segmentos da indústria do software, tal como em Sistemas de Informações Gerenciais (SIG). Muitas das práticas descritas no modelo podem ser diretamente aplicadas ao desenvolvimento de hardware.

O modelo é baseado no CMM versão 1.1. A arquitetura do modelo TRILLIUM difere da versão 1.1 do CMM nos seguintes aspectos:

a) um modelo de arquitetura baseado em guias, em contraposição às áreas chaves do processo;
b) uma perspectiva do produto, melhor que software;
c) extensa cobertura da capacidade de impacto das saídas; e
d) um foco comum, maturidade tecnológica, e uma orientação nas telecomunicações.

O modelo TRILLIUM é baseado ainda nos modelos:

a) ISO 9001: 1994 International standard;
b) ISO 9000-3: guia1991;
c) Bellcore TR-NWT-00^{0179} Issue 2, June, 1993;
d) Bellcore TA-NWT-00^{1315} Issue 1, December, 1993;
e) partes relevantes do Malcolm Baldrige National Quality Award, 1995 Award Criteria;
f) IEC standard Publication 300: 1984.

O TRILLIUM trabalha com uma escala de 1 a 5, para caracterizar as organizações de softwares:

Nível 1 - Não estruturado. Projetos não podem ser qualificados. O sucesso, quando possível, é baseado em talentos individuais, antes da infra-estrutura organizacional.

Nível 2 - Repetitivo e Projeto orientado. O sucesso do projeto é obtido com o controle e planejamento do projeto, com ênfase nos requisitos de gerenciamento, técnicas estimadas e gerenciamento das configurações (risco médio).

Nível 3 - Definido e Projeto orientado. Processos são definidos e utilizados por um nível da organização, embora customização de projetos seja permitida. Processos são controlados e aperfeiçoados. Os requisitos

da ISO 9001, tal como o treinamento e auditorias internas de processo são incorporados (risco pequeno).

Nível 4 - Gerenciado e Integrado. Instrumentação e análise são usados como mecanismos-chaves da melhoria de processos. O gerenciamento da melhoria do processo e os programas de prevenção de defeitos são integrados nos processos. Ferramentas CASE são integradas ao processo (risco muito pequeno)

Nível 5 - Completamente integrado. Metodologias formais são extensamente usadas. Repositórios da organização para histórico do desenvolvimento de processos são utilizadas efetivamente (risco pequeniníssimo).

A arquitetura do TRILLIUM consiste das áreas de Capabilidades, Mapas de atuação e práticas.

3.3.9 SQUID

O projeto Software Quality in Development-SQUID foi iniciado em 1994. É um consórcio de empresa e institutos da Europa. Fortemente baseado nos conceitos das normas ISO 9001, ISO/IEC 9126 e ISO/IEC 14598-3.

O objetivo do SQUID é desenvolver e automatizar modelos, criando um conjunto de ferramentas integradas. Isso possibilita que organizações que desenvolvam software possam monitorar, controlar e predizer a qualidade dos produtos nos vários estágios do ciclo de vida do desenvolvimento de software, com o uso de medições objetivas.

No modelo SQUID, deve-se definir um Modelo de Qualidade e um Processo de Qualidade. O modelo de qualidade deve abordar a visão de qualidade, a visão de dados e a visão de produto. O processo da qualidade divide-se em: especificação, planejamento, controle e avaliação da qualidade.

3.4 CONCLUSÃO DO CAPÍTULO

Desde a programação estruturada da década de 70, vêm sendo estudadas, cada vez mais, as técnicas para o melhoramento da qualidade de produtos que dependem de software. Porém, a teoria é muito diferente da prática das organizações.

Uma pesquisa realizada pelo Software Engineering Institute-SEI, durante os anos de 1992 até 1996, avaliando 533 organizações, revelou que 61,8% destas organizações ainda desenvolvem software, de forma amadora, sem nenhum gerenciamento ou controle, baseando-se apenas na criatividade e heroísmo das pessoas, enquanto apenas 13,5% possuem um processo padronizado e sistematizado de desenvolvimento.

No Brasil, conforme a pesquisa realizada pelo Ministério da Ciência e Tecnologia-MCT, em 1999, com 446 empresas, apenas 3,6% conhece e usa sistematicamente a norma ISO/IEC 12207; esse percentual é menor quando se trata dos modelos CMM e SPICE com 1,8%.

Conforme a SEI (1996), a qualidade e produtividade de uma organização depende de três fatores: processos, tecnologia e pessoas; é o chamado triângulo da qualidade. Qualquer desses fatores estando inadequado, certamente irá comprometer a qualidade e a produtividade de uma organização.

Se o estabelecimento de um processo-padrão em uma organização é elaborado a partir das melhores práticas, é primordial que este trabalho seja realizado integrando pesquisadores e desenvolvedores. Quanto mais distante o relacionamento desse par, mais difícil a internalização do processo-padrão. O treinamento e disciplina do pessoal é fundamental para o entendimento comum das partes envolvidas. O apoio do nível estratégico da organização é essencial para o sucesso do empreendimento; é necessário também um trabalho intenso de auditoria, na fase inicial da implantação.

A partir do momento em que uma organização estabelece um processo padronizado e sistematizado no desenvolvimento de software, cada novo projeto torna-se a consolidação do processo anterior, ganhando em qualidade e produtividade.

Pelas próprias características das empresas desenvolvedoras de software ou de qualquer outra empresa, algumas normas e conceituações dos modelos e normas apresentadas neste trabalho são atendidas, senão em sua totalidade, mas em alguns pontos. A questão reside na forma como essas empresas se situam frente às diretrizes das Normas e Modelos e qual o objetivo de se adequar a um determinado modelo. Tratar a QUALIDADE, no contexto do ambiente de desenvolvimento de software não é uma tarefa fácil.

O Controle da Qualidade de Software, segundo Fernandes (1995), é assim

conceituado:

> **“... Controle da qualidade do software envolve a utilização de técnicas e atividades para o monitoramento dos processos de gestão, desenvolvimento e manutenção/evolução do software, a fim de verificar causas de variabilidade, determinar e eliminar as causas de não-conformidade, de modo a obter-se eficiência econômica no processo.”**

4. TÉCNICAS DE AVALIAÇÃO DE INTERFACES

Avaliação de softwares constitui uma área em que a demanda cresceu significativamente, paralelamente ao estabelecimento de normas e ao desenvolvimento de métodos e técnicas para auxiliar no processo de avaliação. Diversos métodos têm sido desenvolvidos, como ferramentas na obtenção de produtos de software melhores e mais confiáveis. Dentre os tipos básicos existem: a avaliação heurística, teste de usabilidade, conformidade com recomendações e exploração cognitiva.

Pode-se avaliar, seguindo os critérios estabelecidos pelas normas ISO, como uma avaliação geral, complementando com a avaliação de critérios específicos para o contexto de uso do produto de software.

Um critério para classificação e avaliação de software educacional bastante valorizado é o de grau de interatividade conferido a ele. Nesse aspecto, a ergonomia de interface homem-computador fornece a base teórica e metodológica, para estudar as dificuldades de relacionamento homem-máquina.

A ergonomia utiliza métodos e técnicas para observar o trabalho humano, utilizando estratégias para apreender a complexidade do trabalho e decompor a atividade em indicadores observáveis, como a postura, exploração visual e o deslocamento.

Como em todo processo científico de investigação, a espinha dorsal de uma intervenção ergonômica é a formulação de hipóteses. Segundo LEPLAT (in CAMPOS, 2001), o pesquisador trabalha, em geral, a partir de uma hipótese, é isso que lhe permite ordenar os fatos.

Pode-se agrupar as técnicas utilizadas em ergonomia em técnicas objetivas e subjetivas:

a) técnicas objetivas ou diretas: são os registros das atividades ao longo de um período, por exemplo, através de um registro em vídeo;
b) técnicas subjetivas ou indiretas: são aquelas que tratam do discurso do operador, que são os questionários, os *check-lists* e as entrevistas. Esse tipo de coleta de dados pode levar a distorções da situação real de trabalho, entretanto pode fornecer uma gama diferenciada de dados para uma análise preliminar.

4.1 NORMA ISO 9241

A norma ISO 9241 descreve os requisitos ergonômicos para trabalho de escritório com computadores, composta de 17 partes.

Das partes constituintes, as partes 1 a 11 são aquelas que estão em votação na ISO/IEC; e as partes 10 e 11 serão abordadas, por se tratar de atividades diretamente envolvidas com a interface homem computador. A parte 10 refere-se aos princípios de diálogo e a parte 11, às orientações sobre usabilidade. As partes 12 a 17 estão em fase de tradução; tratam da apresentação da informação, das orientações ao usuário e das diferentes formas de apresentação do diálogo.

4.1.1 Princípios de Diálogo

O escopo dessa parte é o projeto ergonômico de software. Escreve os princípios ergonômicos gerais na especificação, projeto e avaliação de diálogos para computadores. Destina-se aos usuários finais, exigindo um envolvimento desses e definindo as responsabilidades sobre a aplicação.(ABNT, 2001).

Os princípios são:

a) adequação à tarefa;
b) autodescrição;
c) controlabilidade;
d) conformidade com as expectativas do usuário;
e) tolerância a erros;
f) adequação à individualização;
g) adequação ao aprendizado.

A aplicação desses princípios:

a) quanto às características dos usuários: observa a capacidade de atenção, memória de curto tempo, capacidade de aprendizado, nível de experiência no trabalho e no sistema e a visão das bases de dados e do sistema;
b) quanto às características da tarefa: descreve o desempenho de uma tarefa condicionado às características do diálogo, por exemplo, por palavra-chave, e à definição do nível de eficácia e eficiência desejados.

A aplicação desses princípios e a importância de cada um, irá variar em função das aplicações e dos usuários e quando da aplicação poderá ser necessário estabelecer

prioridades, com base na análise individual. Os princípios não são independentes entre si, porém, há uma interdependência por afinidades entre eles.

No princípio de adequação à tarefa um diálogo é adequado quando apóia o usuário em uma conclusão efetiva e eficiente da tarefa.

a) convém que o diálogo apresente somente informações relacionadas à conclusão da tarefa;
b) a informação de ajuda (*Help*) seja dependente da tarefa;
c) convém que qualquer ação que pode ser alocada ao software de interface para execução automática deva ser realizada pelo software e não pelo usuário;
d) convém que, ao projetar um diálogo, devam ser feitas considerações sobre a complexidade da tarefa, no que diz respeito às destrezas e habilidades do usuário;
e) convém que o formato de entrada e saída seja apropriado à tarefa dada e aos requisitos do usuário;
f) convém que o diálogo apóie o usuário ao realizar tarefas recorrentes;
g) se existirem recursos de entradas pré-definidas para uma dada tarefa, convém que não seja necessário o usuário entrar com tais valores;
h) ao realizar uma tarefa, na qual os dados são originais, estes devam permanecer acessíveis, se a tarefa o exigir;
i) recomenda-se que o diálogo evite forçar passos desnecessários à tarefa.

No princípio de autodescrição, um diálogo é autodescrito quando cada passo do diálogo é imediatamente compreensível, por meio de respostas do sistema ou é explicado sob demanda ao usuário. As ações devem observar uma área apropriada para respostas do sistema, com uso de terminologia derivada do ambiente da tarefa, servindo como suplemento para o treinamento, adequando-se ao nível de conhecimento que se espera do usuário típico. É preciso que as respostas se refiram à situação à qual sejam necessárias, que a quantidade de explicações e respostas do sistema minimizem a consulta aos manuais. Quando uma entrada é solicitada, convém que o sistema forneça informações ao usuário e que as mensagens sejam formuladas e apresentadas em um estilo compreensivo, objetivo e construtivo, dentro de uma estrutura consistente sem julgamento de valores.

Na controlabilidade, um diálogo é controlável quando o usuário pode iniciar e controlar a direção e o ritmo da interação, até que o objetivo seja atingido. São tratadas ações como a velocidade de interação, continuação do diálogo, retorno ao ponto inicial, interações reversíveis e entradas e saídas controláveis pelo usuário.

O princípio de conformidade com as expectativas diz que um diálogo está em conformidade com as expectativas do usuário, quando é consistente e corresponde, por um lado, às características do usuário, tais como conhecimento da tarefa, educação e experiência, e por outro lado, convenções usualmente aceitas.

Quanto à tolerância ao erro, a norma escreve que um diálogo é tolerante ao erro se, apesar de erros de entrada evidentes, o resultado esperado pode ser obtido, com pouca ou nenhuma ação corretiva do usuário. Descrevem atividades para que a aplicação auxilie o usuário a detectar e evitar erros de entrada, prevenção sobre entradas que causam estados indefinidos ou falhas no sistema de diálogo, oportunidades de cancelamento de ações incorretas, validação e verificação de entradas, antes do processamento, e que a correção de erros seja possível, sem a mudança de estado do sistema de diálogo.

No princípio de adequação à individualização, a norma escreve que um diálogo é capaz de individualização, quando o software de interface pode ser modificado para se adequar às necessidades da tarefa, a preferências individuais e a habilidades do usuário. São observadas as conveniências de fornecimento de mecanismos que permitam que o sistema de diálogo seja adaptado à língua e cultura, conhecimento individual e experiência do usuário no domínio da tarefa, bem como às suas habilidades perceptivas, motoras e cognitivas, e que o usuário possa adicionar comandos, vocabulários e parâmetros operacionais individualizados.

Um diálogo é adequado, quando apóia e guia o usuário, no aprendizado, para usar o sistema. A norma diz que convém que regras e conceitos básicos úteis para o aprendizado estejam disponíveis ao usuário, permitindo-lhe construir suas próprias estratégias e regras de agrupamento, para memorizar atividades; que sejam fornecidas estratégias relevantes para o aprendizado e que forneçam diferentes formas para ajudar o usuário a se familiarizar com os elementos de diálogo da aplicação.

A norma recomenda uma lista de boas práticas para as organizações para a observância dessa parte 10, tais como:

a) elaboração de um guia de estilo: contendo descrição de objetos de interação como menus, janelas, caixas de diálogo, uso do teclado e mouse; sendo específicos, impõem maiores restrições aos projetistas, pois possuem o objetivo de padronizar linhas de produtos;
b) utilização de bibliotecas de classes reutilizáveis: ganha-se em qualidade e produtividade;
c) aplicação de um *checklist* ergonômico no projeto de interface: facilitando a padronização e sistematização de processos de avaliação;
d) desenvolvimento de protótipos com validação junto aos usuários: interação com o usuário aumenta a conformidade com as expectativas e conseqüente aumento da qualidade do produto;
e) avaliação pós-venda;
f) realimentação do processo.

4.1.2 Orientações sobre Usabilidade

A parte 11 da norma ISO 9241 descreve as orientações sobre usabilidade de software, em que inclui orientações sobre como a usabilidade pode ser especificada e avaliada como parte de um plano de qualidade. Descreve as características de:

a) eficácia: grau de realização de objetivos perseguidos na interação;
b) eficiência: recursos alocados para alcançar estes objetivos;
c) satisfação: grau de aceitação do produto pelo usuário.

São descritas as características, envolvendo em seu contexto, os usuários, as tarefas, os equipamentos tanto de hardware, software e documentos, e também o ambiente físico e social. A norma descreve a elaboração do projeto de usabilidade, seu propósito, a ênfase, conforme o ambiente de uso, as medidas e os valores reais ou desejados de eficácia e satisfação, conforme o contexto pretendido e a descrição das partes integrantes do contexto de uso.

As medidas de usabilidade são três, a saber:

a) eficácia: acurácia e abrangência com as quais usuários alcançam objetivos específicos;
b) eficiência: recursos gastos em relação à acurácia e abrangência com as quais usuários atingem objetivos;

c) satisfação: ausência do desconforto e atitudes positivas para com o uso de um produto.

Há uma forte correlação entre as normas ISO 9241 e a ISO/IEC 9126 pela análise das características de um produto, requerida por um contexto particular de uso, de seus processos de interação e pela análise da eficácia e eficiência que resultam do uso e medidas de satisfação no contexto de uso.

A correlação com outras partes da norma refletem na melhoria da interface com o usuário, que são práticas do projeto de diálogo, tratadas na parte 10 e 12 até a parte 17, na aplicação correta de cada princípio de diálogo, tratada na parte 10, e na melhoria dos aspectos do ambiente de trabalho, que é o projeto do posto de trabalho, iluminação, ruído, tratados nas partes 3 até a 9.

As atividades de teste de usabilidade podem ser:

a) exploratórias: em que avaliam a eficiência do conceito preliminar do projeto, realizado no início do ciclo de vida do software;
b) estimativas de qualidade: avaliam-se a usabilidade nas operações de baixo nível e aspectos do produto, podem ser realizadas na fase intermediária do desenvolvimento do produto;
c) de validação: em que avaliam o produto, de acordo com os padrões, realizadas em todas as etapas do ciclo de vida;
d) comparativas: seleção e estabelecimento de qual projeto é mais fácil de usar, realizadas em todas as etapas do ciclo de vida.

4.2 AVALIAÇÃO DO PRODUTO DE SOFTWARE

Diversos métodos de avaliação ampla e geral têm sido desenvolvidos, como ferramentas na obtenção de produtos de software melhores, mais confiáveis e mais amigáveis ao usuário. Dentre os tipos básicos existem: a avaliação heurística, teste de usabilidade, conformidade com recomendações e exploração cognitiva.

Pode-se avaliar, seguindo os critérios estabelecidos pelas normas ISO, como uma avaliação geral, complementando com a avaliação de critérios específicos para o contexto de uso do produto de software. Considerando que os principais fatores de sucesso de um sistema estão, cada vez mais, associados a um convívio adequado deste sistema com os demais sistemas existentes, a visão e a preocupação passa a ser muito

mais organizacional do que técnica, pelo fato de uma organização já estar estruturada em suas necessidades operacionais e demandem em sistemas mais gerenciais e estratégicas. Portanto, em uma avaliação devem ser analisadas as interferências de outros sistemas (políticos, culturais, de poder, econômicos, estilo de gestão), da forma como atuam fortemente sobre os sistemas de informações e esta análise passa a ser determinante na forma do produto de software em sua versão definitiva.

4.2.1 Avaliação Geral usando as normas ISO

Na avaliação geral, o uso da norma ISO/IEC 12119 cobre todos os componentes dos produtos disponíveis ao usuário, tais como a documentação, manual de instruções e guia de instalação. A norma ISO/IEC 9126-NBR 13596 estabelece as características de qualidade de software e a norma ISO/IEC 14598 descreve os processos de avaliação. Desta forma, a avaliação resulta em dados mais confiáveis, mais seguros, permitindo verificar, de forma mais objetiva, a ocorrência e o atendimento aos atributos necessários ao tipo de produto. A avaliação, segundo esses critérios, tem o objetivo de verificar a conformidade com os padrões internacionais na obtenção de softwares mais confiáveis, desenvolvidos em um tempo menor, com processos gerenciáveis em todo o ciclo de vida.

4.2.2 Avaliação Heurística

A avaliação heurística é uma técnica de avaliação baseada em incertezas provisórias, seguindo uma seqüência lógica de passos, realizando aproximações progressivas, com cada caminho percorrido sendo avaliado, e então, especula-se sobre a natureza dos caminhos a seguir, para se aproximar do objetivo de encontrar o maior número possível de problemas de usabilidade. Deve ser realizada por especialista em interface, sendo um método para inspeção de usabilidade, aplicado em sistemas com interfaces gráficas. Em linhas gerais, consiste na realização de uma análise sistemática da interface, para identificação de problemas de usabilidade, tomando como base um conjunto de heurísticas ou princípios de usabilidade (TSUKUMO, 1996).

O processo consiste na identificação de problemas existentes na interface e a importância destes no contexto de uso, fornecendo uma lista de problemas de usabilidade, apresentados por ordem de importância, facilitando o processo de revisão

do projeto da interface. A confiabilidade e a validade da avaliação está diretamente associada à escolha das tarefas e cenários para inspeção, pois é por intermédio delas que a análise se processa e os elementos da interface e a funcionalidade da aplicação é enfocada.

O método heurístico é fundamentado na usabilidade, enfocando a maior facilidade de aprendizagem e utilização do software, de forma indutiva, não necessitando recorrer a qualquer forma de documentação. O avaliador tem maior liberdade e julga, de forma subjetiva, com resultados mais rígidos no domínio do aplicativo (TSUKUMO, 1996).

4.2.3 Teste de Usabilidade

O teste de usabilidade ou utilizabilidade é realizado com a observação da interação de usuários no mundo real ou sob condições controladas. Os avaliadores reúnem os dados dos problemas detectados no uso e verificam se a interface suporta o ambiente e as tarefas do usuário. Os testes realizados em laboratório, sob condições controladas, apresentam a vantagem da existência de maior disponibilidade de equipamentos e de infra-estrutura facilitando as observações, por outro lado, a situação sendo simulada pode não registrar as ocorrências que aparecem em um mundo real. Os testes realizados em campo, em situação real, têm a desvantagem do fator tempo que é maior, porém tem a vantagem das interações entre o sistema e o usuário em seu ambiente natural.

Algumas vantagens do teste de usabilidade, segundo Jeffries (1991):

a) indica as reações dos usuários potenciais ao sistema;
b) mostra os problemas ou as falhas do sistema;
c) mostra onde o sistema funciona bem;
d) ajuda a avaliar as características do projeto e os conflitos;
e) fornece idéias para o projeto, através das sugestões dos usuários;
f) fornece meios para comparação de múltiplos usuários;
g) fornece suporte para um aperfeiçoamento adicional do projeto;
h) promove a participação do usuário.

As desvantagens residem no alto custo, não identificam problemas de consistência e confiabilidade dos dados e a necessidade de especialistas em interface.

Apesar das desvantagens, as vantagens são maiores em qualquer teste de usabilidade, devido ao envolvimento do usuário, na sua participação na avaliação de interfaces. A observação da forma como o usuário realiza as atividades é a característica principal do teste de usabilidade, independentemente da ferramenta de registro utilizada.

4.2.4 Conformidade com recomendações

No uso dessas recomendações, o avaliador ou o próprio projetista verifica a conformidade da interface com as recomendações constantes em guias de recomendações.

O estudo de Jeffries (1991) destacou que esta foi a melhor técnica para a detecção de problemas gerais e repetitivos, concluindo que um conjunto bem selecionado de recomendações serve como um dispositivo focalizador, que força o avaliador a ter uma visão ampla da interface. Também pode ser usado um *checklist* organizado em forma de questões interrogativas, em vez de recomendações na forma afirmativa.

4.2.5 Exploração cognitiva

Na exploração cognitiva, o projetista navega nas funcionalidades, para executar as tarefas principais utilizadas por um usuário típico, com o objetivo de comparar as ações disponíveis e o *feedback* da interface com os objetivos e o conhecimento do usuário.

A simulação da execução das tarefas do usuário pelo projetista, dificilmente consegue ser representativa, devido às diferenças individuais e ao fato de que, normalmente, desconhece a tarefa.

No estudo de Jeffries (1991), esta técnica não apresentou bons resultados, devendo ser aperfeiçoada conforme a constância de uso.

4.3 CONCLUSÃO DO CAPÍTULO

Conforme as normas ISO, o desenvolvimento de um software deve envolver, em todo o ciclo de vida, a avaliação da qualidade, para garanti-la no produto final. Mas apenas a avaliação no contexto de desenvolvimento de processo e produto não garante a sua aplicabilidade em ambiente educacional. É importante que os aspectos do ambiente,

estejam incorporados aos aspectos da adequação da solução com o todo da realidade e necessidade do usuário.

Na história da humanidade, do homem, em sua evolução ao longo do tempo, está sintetizada na busca de formas harmônicas que encerram um ideal de beleza, e a tecnologia persegue este princípio da busca constante, baseada na estética das coisas. Einstein afirmava que toda equação matemática está embasada, inicialmente, em uma manifestação do belo, do qual não podia prescindir. Com a disseminação do computador e com novas ferramentas automatizadas, que permitem a manipulação de inúmeros recursos visuais, manuais, táteis, torna-se mais imperativo que a mediação homem-máquina se articule sobre um arcabouço agradável, aliando a funcionalidade da aplicação à harmonia dos elementos que o compõem, tornando amigável a relação do homem com o meio.

5. TÉCNICAS DE AVALIAÇÃO E DESENVOLVIMENTO DE SOFTWARE EDUCACIONAL

A diferenciação de um software tradicional, de um software com características educacionais, é uma questão problemática, genérica e ampla, bastante complexa e controversa. Quando é que um software torna-se educacional? O software tecnologicamente avançado, com som, animação, cores, profundidade de telas e uma grande variedade de recursos com conteúdo apenas informativo, é um software educacional? Muitos autores propõem a divisão em classes e subclasses, para facilitar a análise e a avaliação de softwares para uso educacional. Na prática, os produtos não funcionam isoladamente, atualmente os sistemas costumam ser híbridos.

A qualidade de software educacional deve ser determinada pelas teorias de aprendizagem, que distinguem ambientes educacionais, mais ou menos interativos, com maior ou menor grau de participação e controle do aluno no processo de construção do conhecimento. O desenvolvimento de um software educacional, portanto, deve contemplar as características da educação, que levam à formação global do aluno, que necessita aprender a aprender e a pensar para melhor intervir, inovar e questionar, deve trabalhar com as funções de cognição do aluno (CAMPOS, 2001).

Um critério para classificação e avaliação de software educacional bastante valorizado é o de grau de interatividade conferido a ele. Nesse aspecto, a ergonomia de interface homem-computador fornece a base teórica e metodológica para estudar as dificuldades de relacionamento homem-máquina. A ergonomia utiliza métodos e técnicas para observar o trabalho humano, utilizando estratégias para apreender a complexidade do trabalho e decompor a atividade em indicadores observáveis, como a postura, exploração visual e o deslocamento. Como em todo processo científico de investigação, a espinha dorsal de uma intervenção ergonômica é a formulação de hipóteses. Segundo LEPLAT (in CAMPOS, 2001), o pesquisador trabalha, em geral, a partir de uma hipótese, é isso que lhe permite ordenar os fatos.

Cabe aqui analisar como a informática vem sendo utilizada no processo de ensino-aprendizagem, como a tecnologia interfere no processo cognitivo; quais os critérios estabelecidos para as aplicações serem consideradas educacionais; quais os seus pressupostos pedagógicos; cumpre os critérios da *ecologia cognitiva*?

O termo ecologia cognitiva foi cunhado recentemente para indicar a

importância que as tecnologias têm no processo cognitivo. Para Lévy (1993), as sociedades orais dependem apenas do que pode ser inscrito na mente, tendo quase todo o seu edifício cultural fundado sobre as lembranças, concluindo-se que a inteligência se associa à memória, estando mais embasada na intuição e compreensão do que na explicação. Para Lévy (1993), as formas de representação que mais chance têm de sobreviver à memória humana, nessa sociedade, são os mitos, as narrações, as danças, as rimas e os cantos, usando as melhores estratégias de codificação culturais. Paulo Freire (1987), no entanto, diz que sem escrita não há datas, nem arquivos, nem códigos legislativos, nem sistemas filosóficos, nem diagramas, concluindo que a palavra escrita é o elemento constitutivo indispensável ao pensamento humano.

A tecnologia por si só não é um instrumento para resolver todos os males, mas auxilia no processo de cognição, usando tanto a escrita como a oralidade das soluções, criando novos ambientes, com grandes possibilidades de serem explorados; a esses ambientes chamamos de ecologia cognitiva, em que convivem as tecnologias de informação, com as engenharias cognitivas e pedagógicas e principalmente com o elemento humano e suas interações.

5.1 DEFINIÇÃO

Quando um software é considerado educacional?

Para Chaves (2001), quando é usado para um objetivo educacional ou pedagogicamente defensável, qualquer que seja a natureza ou finalidade para a qual tenha sido criado.

Para Galvis (1999), ambientes educativos computadorizados que oferecem muito mais do que uma boa utilização educativa de soluções informáticas genéricas, são os materiais educativos computadorizados que, por sua própria natureza, são desenhados para formar o marco do aprendizado e que foram elaborados com uma finalidade educativa específica. Não basta fazer um software com orientação educativa, ou ter brilhantes idéias educativas e dar-lhes sustentação informatizada, deve ser de natureza interdisciplinar, em que o educativo é um dos eixos, o informático o outro e o comunicacional o terceiro eixo.

Para Campos (2001), um software educacional contempla as características da educação que levam à formação global do aluno, que necessita aprender a aprender e a

pensar para melhor intervir, inovar e questionar, trabalhando com as funções da cognição.

Para Vieira (2001), um software para ter uso educacional, deve ajudar o aprendiz a construir seu conhecimento e a modificar sua compreensão de mundo, elevando sua capacidade de participar da realidade que está vivendo, dentro de uma proposta pedagógica.

Costa (in Vieira, 2001) afirma que depende da forma como o usuário final foi considerado na concepção do software, na sua forma de aprender e os objetivos de aprendizagem que se pretende alcançar. Deve-se verificar se a aplicação está clara e explicitamente estruturada, com base num determinado modelo didático e de aprendizagem, que forma assumem os diferentes componentes e qual o potencial para uma utilização com fins educativos.

Nessas definições, o parecer enfoca pontos em comum, que a abordagem clássica de engenharia de software não é apropriada para construção de ambientes de desenvolvimento para programas educacionais, pois, além de absorver todos os componentes de um produto de software tradicional, deve envolver equipes de desenvolvimento multidisciplinar, devendo refletir os objetivos educacionais propostos e a ecologia cognitiva a que se propõe, criando situações que estimulem o desenvolvimento das habilidades desejadas.

Paulo Cysneros (2001) classifica o software educativo em duas grandes categorias: a transposição para o computador de formas tradicionais de ensinar e a aplicação dos recursos inerentes à ferramenta de ensino e à aprendizagem de conteúdos específicos. Na primeira categoria, em que tem ocorrido a maior produção, é a transposição de conteúdos impressos sem grandes mudanças, constituindo-se em exemplos de modernização conservadora, enriquecidos superficialmente com movimentos, imagens, sons, cores vivas, apresentação gráfica atraente, em que o conteúdo e a forma de ensinar permanecem inalterados, apesar de maquiados. Na segunda categoria, produz-se um bom software educativo, apresentando problemas como a manipulação do conteúdo diferenciado do livro didático ou da aula convencional, questionando, por exemplo, se o uso de tutoriais (mesmo os tutoriais considerados inteligentes) é solução desejável e adequada, pois pode limitar a criatividade e a participação ativa do aprendiz.

5.2 FERRAMENTAS DE AVALIAÇÃO

Algumas técnicas, como o sistema Microsift e o NCET CD ROM, os critérios descritos por Bitter & Wighton (1987), bem como os outros tratados no TICESE (GAMEZ, 1999), são *checklists* que contribuem grandemente para o estado em que se encontram as técnicas de avaliação de softwares educacionais, especialmente no mercado brasileiro.

Segundo Heemann (1997), *checklist* é uma ferramenta para avaliação da qualidade ergonômica de um software, que se caracteriza pela verificação da conformidade da interface de um sistema interativo com recomendações ergonômicas provenientes de pesquisas aplicadas. O uso de instrumento como um *checklist* genérico pode levar a resultados que não condizem com a realidade da aplicação, a forma de resolver este problema seria o desenvolvimento de um *checklist* especializado mais adequado ao ambiente de uso do software.

5.2.1 TICESE

Técnica de Inspeção de Conformidade Ergonômica de Software Educacional-TICESE, elaborada a partir da revisão dos vários modelos de avaliação de software propõe a integração entre os aspectos pedagógicos e de usabilidade no processo de avaliação ergonômica do software educacional (LABUTIL, 2000).

A técnica é composta por um conjunto de critérios e subcritérios de inspeção, baseados em recomendações ergonômicas para o serviço de avaliação. É formada por 3 seções, formando o Manual do Avaliador: a seção 1 contém a Descrição e Justificativas dos Critérios, a seção 2, as diretrizes para o Tratamento Quantitativo da Informação e a seção 3, o Formulário de Inspeção Ergonômica de Software Educacional, complementados por dois módulos introdutórios: o Módulo de Classificação, com o objetivo de classificar, em uma proposta pedagógica, dentro de atributos específicos de cada categoria de software, e o módulo de avaliação que constitui o módulo principal da técnica em que se avalia a conformidade com os padrões ergonômicos de qualidade de software. Dentre os critérios de apontados estão:

a) na avaliação da documentação: dados de identificação, qualidade da informação impressa, presteza, legibilidade, densidade informacional, consistência e significado dos códigos e denominações;

b) na avaliação do produto: condução, presteza, qualidade da opção de ajuda, legibilidade, feedback imediato, agrupamento e distinção de itens por localização e por formato, adaptabilidade, observando a flexibilidade e consideração da experiência do usuário, controle explícito, recursos de apoio à compreensão dos conteúdos, gestão de erros, avaliação da aprendizagem, carga de trabalho, significado dos códigos e denominações, homogeneidade e compatibilidade.

No módulo específico denominado Módulo de Avaliação Contextual, verifica-se a adequabilidade do software em um dado contexto pedagógico ou situação específica. Sendo um módulo complementar na avaliação, auxiliando no processo de tomada de decisão sobre a adoção e implementação em um contexto particular de ensino.

A seção dois, em que se trata de propostas de diretrizes para o tratamento quantitativo da informação é composta das seguintes atividades:

a) definição do sistema de valores para tratamento quantitativo dos dados;
b) definição do sistema de ponderação para as respostas ao formulário de inspeção;
c) definição de pesos a atribuir para cada questão;
d) definição de uma equação matemática para o tratamento da informação; e
e) aplicação do sistema de métrica e interpretação dos resultados.

5.2.2 Pesquisa com professores

Esta pesquisa foi baseada em um artigo da pesquisadora Gilda Helena B de Campos, publicada no site da Casa da Ciência sobre Qualidade em Software Educacional, em que se relata a pesquisa realizada com professores do Estado do Rio de Janeiro sobre os critérios a serem adotados em uma avaliação da qualidade de um software educacional (CAMPOS, 2001). Esse documento originou o seguinte *checklist*:

- possibilidade de correção de conteúdo (alterabilidade);
- facilidade de leitura da tela (amenidade ao uso);
- clareza de comandos (amenidade ao uso);
- independência da linguagem e do hardware (independência do ambiente);
- adaptabilidade ao nível do usuário (eficiência do processamento);

- adequação do programa ao nível do usuário (validabilidade);
- facilidade de leitura do programa (clareza);
- ausência de erros no processamento do programa (correção);
- adequação do programa às necessidades curriculares (rentabilidade);
- existência de recursos motivacionais (amenidade ao uso);
- previsão de atualizações (validabilidade);
- ausência de erros de conteúdo (validabilidade);
- possibilidade de inclusão de novos elementos (alterabilidade);
- resistência do programa a respostas inadequadas (robustez);
- adequação do vocabulário (amenidade ao uso);
- fornecimento de *feedback* (amenidade ao uso);
- apresentação dos escores aos alunos (validabilidade);
- uso do tempo do equipamento (rentabilidade);
- integração do programa com outros recursos (rentabilidade);
- capacidade de armazenamento das respostas (eficiência do processamento);
- existência de tratamento de erro (amenidade ao uso);
- diagramação de telas (amenidade ao uso);
- tempo de resposta (eficiência do processamento);
- existência de ramificações para enfoques alternativos (amenidade ao uso);
- existência de mensagem de erro (amenidade ao uso);
- acesso a *helps* (amenidade ao uso);
- uso de cor (amenidade ao uso);
- tempo de exposição de telas (amenidade ao uso);
- uso de animação (amenidade ao uso);
- existência de geração randômica de atividades (amenidade ao uso); e
- uso de recursos sonoros (amenidade ao uso).

Os professores privilegiaram critérios sobre situações importantes para a prática pedagógica do cotidiano, evidenciando uma supremacia da estrutura pedagógica e da construção de conhecimentos nos critérios de avaliação selecionados. Não se discutem aqui as tecnologias de construção de interfaces e nem a ecologia cognitiva, falta o fator USUÁRIO nos critérios estabelecidos.

A pesquisadora estabelece uma correlação entre a extinção das matérias de

filosofia e psicologia na reforma pedagógica de 1968, dos currículos regulares de ensino do segundo grau, instante esse em que o professor perdeu o vínculo com o seu trabalho de educador, tornando-se um professor descartável, uma vez que o conteúdo das suas aulas era um conteúdo padrão, ministrável em qualquer estabelecimento de ensino do país, com o desenvolvimento de técnicas de avaliação de qualidade de software educacional privilegiando a necessidade de se resgatar uma certa postura pedagógica em que o professor volte a construir o conteúdo, a desenvolver habilidades cognitivas, raciocínio lógico, pensamento e escrita e o poder de decisão, tornando-os requisitos fundamentais para avaliação da qualidade.

5.2.3 Modelo Escola Internet

Esta técnica foi baseada na coluna de treinamentos on-line do site da Escola Internet, mantida pela professora Gilda Campos, sobre qualidade de software educacional, o qual parte de um artigo anterior desenvolvido e já publicado em parceria com a professora Fernanda Campos da Universidade Federal de Juiz de Fora (MG) (CAMPOS, 2001).

Conforme a professora Gilda Campos, o software educacional deve ser determinado pelas teorias de aprendizagem, que distinguem ambientes educacionais, mais ou menos interativos, com maior ou menor grau de participação e controle do aluno no processo de construção do conhecimento. Nos ambientes de aprendizagem construtivista, a ênfase está na autonomia do aluno, que interage com o ambiente, e o foco está no processo de construção do conhecimento e não apenas em um domínio pré-definido do conhecimento a ser adquirido.

Um critério muito usado para classificar os tipos de software educacional é o grau de iniciativa permitido ao aluno ou o grau de direcionamento conferido a ele:

a) alta interatividade: permite a descoberta imprevista e a descoberta com exploração livre;
b) média interatividade: permite a descoberta guiada;
c) baixa interatividade: privilegia a aprendizagem de recepção direcionada, a exposição indutiva, a exposição dedutiva.

O modelo construtivista de software é caracterizado por uma definição de objetivos macros e os contextos para incentivar a construção do conhecimento com a

participação do aluno no processo, a avaliação é qualitativa, considerar a não-linearidade, a escolha de caminhos de navegação por parte do aprendiz e a liberdade na busca da informação, propor problemas realistas, interessantes e relevantes para os alunos e permite testar diversas soluções e por estimular a colaboração, o diálogo e a negociação no trabalho em grupo.

Para avaliar um software educacional é preciso considerar, além das características citadas, os atributos inerentes ao domínio e às tecnologias específicas, bem como as diversas visões das teorias de aprendizagem e cognição.

Os parâmetros mínimos de avaliação referem-se às características pedagógicas e ao aspecto técnico da questão:

a) características pedagógicas: formam um conjunto de atributos que evidenciam a conveniência e a viabilidade de uso em situações educacionais. Tem como subcaracterísticas: o ambiente educacional identificando o ambiente e o modelo de aprendizagem, a pertinência em relação ao programa curricular em que analisa a adequação em relação ao contexto educacional ou a uma disciplina específica, e também os aspectos didáticos que analisa a contribuição para que o aluno alcance o objetivo e os meios motivacionais respeitando a individualidade;
b) facilidade de uso: conjunto de atributos que evidenciam a usabilidade de software. Inclui as subcaracterísticas como: facilidade de aprendizado, facilidade de memorização e robustez em que analisa se o software mantém o processamento corretamente a despeito de ações inesperadas;
c) características de interface: avaliam os atributos que evidenciam a existência de meios e recursos que facilitam a interação do usuário com o software. Inclui as subcaracterísticas:
    - de condução com atributos como presteza, localização, *feedback* imediato e legibilidade,
    - de afetividade que avalia se o software proporciona uma relação agradável com o aluno ao longo do processo de aprendizado,
    - de consistência que avalia se a concepção da interface é conservada igual em contextos idênticos, e se ela altera em contextos diferentes;
    - significado de códigos e denominações: avalia a adequação entre o objeto

ou informação apresentado ou pedido e sua referência;

- gestão de erros: avalia os mecanismos que permitem evitar ou reduzir a ocorrência de erros, que favoreçam a correção. Inclui atributos como proteção contra erros, qualidade das mensagens de erro e correção dos erros e reversão fácil das ações;

d) adaptabilidade: avalia a capacidade do software de adaptar-se às necessidades e preferências do usuário e ao ambiente educacional selecionado. Inclui atributos como a customização, em que avalia a facilidade de adaptação da interface para diferentes usuários e a adequação ao ambiente que avalia a facilidade de adequação do software aos objetivos educacionais;

e) documentação: avalia se a documentação está completa, consistente, legível e organizada. Inclui atributos como *help on line,* em que avalia a existência de auxílio *on line* e a documentação do usuário em que avalia a documentação sobre o uso do sistema, sobre a instalação e sobre a clareza de compreensão.

A autora destaca as características para ambientes e sites educacionais apoiados na WEB, incluindo subcaracterísticas como conteúdos corretos, fontes fidedignas, carga informacional compatível, pertinência, temas transversais, entre outros.

5.2.4 Modelo Planeta Educação

Este site oferece aos usuários, na coluna “Consulte um especialista”, dicas úteis para análise de softwares educacionais, com uma proposta de modelo em forma de *checklist,* com gradação de valores de 1 a 5, em que o 5 é o maior valor, com conceito “muito bom”, e o 1, o menor, com conceito “ruim” (PLANETA EDUCAÇÃO, 2001).

PONTUAÇÃO	CONCEITO
45 a 50	Excelente
45 a 44,5	Ótimo
30 a 39,5	Bom
20 a 29,5	Regular
10 a 19,5	Ruim
0 a 9,5	Péssimo

O controle de pontuação é feito estabelecendo o peso 1,5 aos itens de 01 a 07 e peso 0,5 aos itens de 08 a 10, somando os pontos e consultando-se a tabela de classificação, conforme demonstra o esquema apresentado.

O *checklist* apresenta a parte 1 com as perguntas:

1) apresentação gráfica (cor, imagens, simbolismo, qualidade do grafismo e outros);
2) qualidade dos sons (adequados à temática: clareza, riqueza, qualidade do som);
3) linguagem utilizada (riqueza e adequação à idade, com gírias e/ou vocabulário regionalizado);
4) interação do aluno (pouco/muito interativo, interatividades superficiais ou não, se adequada ou não ao tema e/ou idade, chances de continuidade, *feedback*);
5) propostas de realização de trabalho (qualidade das propostas de trabalho, relacionadas ou não com a temática do software: pintura, jogos, histórias, pesquisa);
6) proposta pedagógica (conciliação entre motivante, divertido e produtivo, adequado à idade, procura ou não propiciar aprendizagem, crescimento e interação, faz o aluno pensar, pesquisar, navegar, dá respostas prontas);
7) temática (adequação à idade proposta pelo fabricante, significativa, relevante à cultura).

Parte 2 – Avaliação Técnica

8) velocidade (ocupação de memória, velocidade de transição de tela);
9) manual (fácil compreensão, detalhado, explicativo);
10) conflito (quando instalado e/ou removido, gera conflito na máquina como arquivos indesejáveis que não são removidos).

Nesta avaliação, privilegiam-se os aspectos pedagógicos, atribuindo-lhes peso 1,5; enquanto aos aspectos técnicos atribui-se o peso 0,5. Também não se podem atribuir conceitos de avaliação para as subcaracterísticas individualmente, pode-se analisar de uma forma genérica cada item, o que não contribui para um avaliação de qualidade.

5.2.5 Avaliação Segundo a Concepção Construtivista de Aprendizagem

Essa proposta está baseada na idéia de que o uso do computador na educação tem o objetivo de promover a aprendizagem dos alunos e ajudar na construção do processo de conceituação e no desenvolvimento de habilidades importantes para que ele participe da sociedade do conhecimento e não simplesmente facilitar o seu processo de aprendizagem (VIEIRA, 2001).

Avaliar significa analisar como um software pode ter um uso educacional, como pode ajudar o aprendiz a construir seu conhecimento e a modificar sua compreensão de mundo, elevando sua capacidade de participar da realidade que está vivendo. Considerando desta forma, uma avaliação criteriosa pode apontar para que tipo de proposta pedagógica poderá ser mais bem aproveitado.

A autora diz que a primeira tarefa de um avaliador de software educacional é identificar a concepção teórica da aprendizagem que o orienta. Numa perspectiva construtivista, a aprendizagem ocorre quando a informação é processada pelos esquemas mentais e agregada a esses esquemas. Assim, o conhecimento construído vai sendo incorporado aos esquemas mentais que são colocados para funcionar diante de situações desafiadoras e problematizadoras.

Um software em uma concepção construtivista deve ser definido dentro de um ambiente interativo, que proporcione ao aprendiz investigar, levantar hipóteses, testá-las e refinar suas idéias iniciais, construindo o seu próprio conhecimento.

Para Valente (in VIEIRA, 1998), a realização do ciclo descrição – execução – reflexão – depuração – descrição é de extrema importância na aquisição de novos conhecimentos por parte do aprendiz. E esse ciclo só será possível se for mediado pelo "agente de aprendizagem", que tenha conhecimento do significado do processo de aprender por intermédio da construção do conhecimento.

A autora classifica os softwares educacionais de acordo com seus objetivos educacionais em: tutoriais, programação, aplicativos, exercícios e práticas, multimídia e internet, simulação e modelagem e jogos. Também, classifica-os quanto ao nível de aprendizagem:

a) seqüencial: a preocupação é transferir a informação; o aprendiz recebe a informação, memoriza e repete quando solicitado, levando a um aprendiz passivo;

b) relacional: objetiva a aquisição de algumas habilidades, permitindo que o aprendiz faça relações com outros fatos ou outras fontes de informação. Esse nível de aprendizagem leva a um aprendizado isolado;
c) criativo: associado à criação de novos esquemas mentais, possibilita a interação entre pessoas e tecnologias, compartilhando objetivos comuns. Esse nível de aprendizado leva a um aprendiz participativo.

Além da base pedagógica, avaliam-se também os aspectos técnicos orientando-os para uma adequada utilização. Analisam aspectos de mídias empregadas, qualidade de telas, interface disponíveis, clareza de instruções, compartilhamento de rede local e internet, compatibilização com outros softwares, hardware e funcionalidade em rede (importação e exportação de objetos), apresentação auto-executável, recursos *hipertexto* e *hiperlink*, disponibilidade de *helpdesk*, manual técnico com linguagem apropriada ao professor-usuário, facilidade de instalação, desinstalação e manuseio, entre outros.

O modelo utiliza-se de uma ficha de registro contendo:

a) identificação do software: autor, firma desenvolvedora, objetivo, breve resumo, idioma, duração, preço e o meio de armazenamento;
b) base pedagógica: a concepção teórica da aprendizagem, se construtivista ou behaviorista com justificativa, a possibilidade do ciclo de Valente, a interação aprendiz-agente de aprendizagem, o *feedback*, o processo de construção do conhecimento do aluno, a formulação e verificação de hipóteses, análise e depuração dos resultados e a interdisciplinaridade;
c) classificação: tutorial, exercícios e prática, programação, aplicativo, multimídia, sistema de autoria, simulação, modelagem, jogos e se aberto ou fechado; quanto ao nível de aprendizado, seqüencial, relacional ou criativo;
d) aspectos técnicos: a presença ou não de alguns aspectos já citados acima;
e) conclusões: sobre o processo de avaliação, recomendações, sugestões e equipe avaliadora.

A autora conclui o seu trabalho afirmando que quem determina as possibilidades de uso dos softwares na educação são os professores, com suas concepções sobre o que é ensinar e aprender.

5.2.6 Avaliação e melhoria da qualidade de SE

Este método foi desenvolvido pelo Centro Tecnológico para Informática-CTI, atualmente ITI, ligado à Universidade Estadual de Campinas, em São Paulo. É um projeto de avaliação fundamentado no método de avaliação de software, chamado Programa de Qualidade e Produtividade em Software-PQPS, utilizando os modelos de qualidade de produto, usando como objeto da análise o software "Escritor", desenvolvido no Laboratório de Educação e Informática Aplicada-LEIA da Faculdade de Educação da Unicamp (CTI, 2001).

O Escritor é um software destinado a ser usado dentro do contexto da escola como instrumento auxiliar para o processo de alfabetização. É um processador de textos simplificado, desenvolvido com o objetivo de facilitar a produção da escrita daqueles que fazem suas primeiras incursões no mundo da escrita, tanto para adultos como crianças, oferecendo um ambiente amigável e fácil de interagir. Destina-se também à educação especial e aos iniciantes em informática, possuindo as funções básicas de um processador de textos comercial.

A avaliação foi fundamentada em duas grandes áreas: avaliação geral e a avaliação heurística. Na geral, foi avaliado o produto como um todo, utilizando critérios baseados nas normas ISO 12119 e a 9126, incluindo a verificação da documentação, instalação do produto e execução do produto com base nas características de qualidade. Na análise, abordaram-se três componentes: software, interface e documentação. Para o software, avaliou-se as características de funcionalidade, eficiência, portabilidade e confiabilidade. Para a interface e documentação, examinou-se a funcionalidade e usabilidade.

A avaliação heurística foi realizada em duas fases, sendo a primeira a avaliação feita sobre o produto de forma livre, sem roteiros ou tarefas específicas, com o objetivo de compor uma visão geral da aplicação. Na segunda fase, as descrições das tarefas foram entregues e os avaliadores realizaram atividades típicas de professores e alunos, anotando os comentários, observações e sugestões dos avaliadores, durante a seqüência das operações do aplicativo.

Concluiu-se que, a avaliação geral, sendo mais abrangente, permite a realização da avaliação das características de qualidade do produto como um pacote de software, incluindo documentação e a identificação dos componentes do produto.

E o método heurístico, sendo fundamentado na usabilidade de software, enfoca

a maior facilidade de aprendizagem e utilização do produto. de forma indutiva, não recorrendo à documentação. Sendo avaliados itens como botões, menus, interação com o usuário, facilidade de apresentação.

Os métodos de avaliação geral e o método heurístico mostraram-se complementares, com a convergência de resultados quanto às características de usabilidade do software.

5.2.7 Método Pedactice

Método desenvolvido pela Universidade de Lisboa, apresentado no 1º Simpósio Ibérico de Informática Educativa em Aveiro, em setembro de 1999, por Fernando Albuquerque Costa, sob o título “Contributos para um Modelo de Avaliação de Produtos Multimedia Centrado na Participação de Professores” (COSTA, 1999).

A avaliação está centrada nas características do produto que permitam reflexão sobre o potencial para exploração pedagógica e que permita a familiarização dos professores com os produtos, envolvendo-os no processo de exploração e reflexão sobre as potencialidades, a adequação ao currículo ou de promoção da aprendizagem. Examinam-se as características que possam ser entendidas como recurso de aprendizagem, principalmente as que mais diretamente possam constituir suporte para a aprendizagem e recurso pedagógico para os professores e não apenas a análise dos aspectos técnicos de construção (*design* e realização).

O modelo de análise é dividido em quatro grandes grupos: o primeiro grupo, a descrição dos requisitos técnicos; o grupo dois descreve os requisitos para análise de conteúdo da aplicação e os aspectos pedagógicos; o grupo três, as interfaces gráficas, a interatividade e as ferramentas de exploração; o grupo quatro, especificamente a usabilidade do software.

a) Grupo I: examinam-se as características do equipamento requerido, a informação técnica sobre o software e o processo de instalação.
b) Grupo II:
   - conteúdo da aplicação: examinam-se as subcaracterísticas: de conteúdo científico, de conteúdo sociocultural, étnico e ideológico, de conteúdo pedagógico, da estrutura e organização da informação, de extensão e densidade da informação e de domínio e nível de complexidade do

conteúdo;

- aspectos pedagógicos: avaliam o público a que se destina, o contexto curricular de uso, os objetivos de aprendizagem, as estratégias de exploração da informação, a motivação, a autonomia na aprendizagem, a interação social e as formas e instrumentos de avaliação;

c) Grupo III:
   - interface gráfica: avaliam as zonas de comunicação e as formas de representação da informação;
   - interatividade: avaliam-se a estrutura de comunicação, o *feedback* e o grau de participação e controle pelo usuário;
   - ferramentas de exploração: mecanismos de ajuda, meios de navegação, sistema de orientação, registro de notas e impressão e exportação da informação;

d) Grupo IV: avaliam-se as subcaracterísticas de necessidade, utilidade, flexibilidade, versatilidade, viabilidade e solidez, facilidade de aprendizagem, valor atribuído ao conteúdo, satisfação global, documentação de apoio e a avaliação global sobre a validade como ferramenta de aprendizagem.

Para cada grupo há uma descrição dos objetivos a serem alcançados em cada subcaracterística e a justificativa para a sua avaliação. A avaliação é do tipo heurística, sem o estabelecimento de valores concretos e, sim, uma valoração subjetiva, com questões sobre o plano descritivo e o plano valorativo subdivido em questões, critérios e indicadores. Os indicadores incluem as características já amplamente divulgadas e validadas pelos seus autores descritos em cada avaliação, além das observadas e trabalhadas no desenvolvimento do projeto.

Incluiu-se também um espaço para feedback ao projeto de avaliação, para sugestões e propostas para cada item de análise.

Essa proposta é bastante extensa, fundamentada em questões relativas aos atributos de cada grupo, enfocando os pontos a serem avaliados em um software educativo, mas trata superficialmente cada item proposto, com formulação de questões a serem analisadas com critérios comentados, quanto aos indicadores são úteis a medida que o avaliador busque o conteúdo de cada um para embasar a sua avaliação. Uma

complementação desse modelo com a visão do usuário completaria o ciclo avaliativo de um software educacional.

5.3 MODELOS DE DESENVOLVIMENTO DE SOFTWARE EDUCACIONAL

O desenvolvimento de um software educacional depende da formação de uma equipe multidisciplinar, envolvendo programadores, psicólogos, pedagogos, cognitivistas e projetistas de interfaces, isso reflete a conclusão unânime dos modelos estudados. Complementa-se com alguns autores admitindo a participação do usuário nessa equipe.

A dúvida, entretanto, reside na definição da fronteira entre o que é software e o que é ambiente educacional. Um site informativo é um software educacional? Uma enciclopédia em CD-ROM é um software educacional? Essas reflexões levaram a uma classificação em algumas categorias, de acordo com os objetivos pedagógicos e as características técnicas de construção: tutoriais, programação, aplicativos, exercícios e práticas, multimídia e internet, simulação e modelagem e jogos.

a) tutoriais: funcionam como um guia, um livro animado, um vídeo interativo ou um professor eletrônico. A informação é apresentada seguindo uma seqüência deixando a escolha para o usuário. Não há um *feedback* da aprendizagem, o controle é do usuário e a interação se resume em avançar ou retroceder dentro da informação apresentada na tela e na sua leitura. É uma versão sofisticada da instrução programada, com recursos de animação, com a vantagem de o usuário programar a sua aprendizagem. Os bons tutoriais utilizam as técnicas de inteligência artificial para analisar padrões de erro, avaliar estilo e a capacidade de aprendizagem do usuário;
b) programação: permite que os usuários criem seus próprios programas, sem o conhecimento de linguagem ou técnicas de programação. A linguagem LOGO é um exemplo deste tipo de software. O usuário aprende a aprender, utilizando o ambiente para espelhar seus conhecimentos e construir novos objetos;
c) aplicativos: são considerados os softwares de produtividade, como os processadores de texto, planilhas eletrônicas, gerenciadores de banco de dados, softwares gráficos. São de uso restrito, mas podem ser explorados

utilizando-se de macrocomandos embutidos nos pacotes. Destinam-se a uma atividade com conteúdo preciso, como esquematizar, classificar objetos ou resolução de problemas numéricos;

d) exercícios e prática: são do tipo pergunta e resposta, com a avaliação monitorada pelo próprio computador. As atividades exigem apenas o fazer, o memorizar informação, não analisando a compreensão e nem a avaliação da aprendizagem. É considerada a forma mais comum de uso do computador na educação. São utilizados para revisar o material visto em sala de aula, especialmente aqueles que exigem memorização e repetição. São indicados para o próprio aluno avançar no conteúdo ou para auxiliar os mais atrasados, com dificuldade de acompanhamento;

e) multimídia e internet: o uso de multimídia já pronta e internet são parecidos com os tutoriais, com a diferença que o usuário pode interagir, navegando com mais facilidade e refletir sobre o conteúdo, mas não oferece oportunidade de compreensão e a aplicação da informação recebida. Já os sistemas de autoria são sistemas em que o usuário pode desenvolver o sistema de multimídia, selecionando as diversas fontes e programas, construindo a sua aprendizagem, possibilitando reflexões sobre os resultados obtidos, comparando-os e depurando a qualidade, profundidade e o significado da informação apresentada;

f) simulação e modelagem: implicam a criação de modelos simplificados do mundo real e são riquíssimos, pois envolvem modelos de sistemas complexos e dinâmicos. As simulações oferecem ao aprendiz a possibilidade de desenvolver hipóteses, testá-las, analisar resultados e refinar os conteúdos, favorecendo o trabalho em grupo. A interação com o computador é do controle do aprendiz. Esse tipo de software favorece o uso do computador e facilita o trabalho de ensino do professor, pois possibilita que situações difíceis e perigosas sejam reproduzidas em aula, como laboratórios virtuais de química, física, biologia, balística, dissecação de cadáveres e outros, apenas simulando a realidade virtualmente;

g) jogos: os jogos pedagógicos têm como alvo a promoção da aprendizagem. A pedagogia, por trás desta modalidade, é a exploração autodirigida, em vez

da instrução explícita e direta. Há alguns desvios no seu uso, devido à competição, como em qualquer jogo, desviando a atenção em relação ao conceito pedagógico para o objetivo de vencer no jogo. Há também a questão motivacional, após vencido o jogo, a repetição torna-se maçante e o aprendiz perde o interesse.

Segundo Galvis (1999), a qualidade do software não se limita ao cumprimento dos padrões associados ao tipo ou combinações de tipos aos quais pertence. Acima disso, estão os critérios de pertinência, relevância (até onde o software é coerente com os outros elementos do ambiente de aprendizagem?) e unicidade (em que medida são aproveitadas as qualidades únicas do computador como meio?). Assim a qualidade é algo que não só está ligada ao produto, mas sim enraizada em todo o processo de desenvolvimento.

5.3.1 Diretrizes para Interfaces Educacionais

Esse modelo propõe a discussão do projeto de interfaces com o usuário para ambientes educacionais, analisando-se a literatura sobre a interação homem-máquina e sobre as interfaces educacionais, com o propósito de definir algumas diretrizes no desenvolvimento de projeto de interfaces, sob a ótica do projetista e não do usuário (SANTOS, 2001).

As diretrizes que nortearão os projetos de interfaces dos protótipos educacionais são:

a) utilizar, tanto quanto possível, uma abordagem conceitual: seja apoiada em enfoques teóricos, em aprendizagem baseada em problemas, aprender fazendo, aprender explorando e navegando, para encontrar respostas, em metáforas do mundo real ou em narrativas e histórias;

b) integrar as tecnologias disponíveis no projeto de interface: a partir da abordagem conceitual escolhida. Narrativas e histórias podem ser mais bem representadas em interfaces com componentes hipermídia e com espaços de comunicação e cooperação. A adoção de uma metáfora do mundo real pode ser mais bem viabilizada como realidade virtual. Aprender resolvendo problemas, explorando e navegando, pode ser mais bem resolvido com interfaces para a Web, pois esses enfoques podem requerer acesso a

múltiplas fontes de consulta;

c) dar ao usuário a possibilidade de adequar a interface a suas necessidades: através dos níveis mais simples e fáceis de customização, como a definição de preferências e a escolha de modelos;
d) considerar, tanto quanto possível, a questão cultural no projeto da interface: observando duas vertentes: a primeira, soluções padronizadas, como as usadas no padrão Windows, podem ser adotadas, já que grande parte dos usuários de software está familiarizada com a metáfora, com os ícones e com as formas de trabalho empregadas; a segunda, são soluções baseadas nas especificidades culturais;
e) considerar na interface educacional os princípios gerais para projeto de interfaces, tais como: funcionalidade, confiabilidade, disponibilidade, segurança, integridade dos dados, padronização, integração, portabilidade, consistência, integridade estética, estabilidade, estilo de *input see-and-point*, manipulação direta, uso de *feedbacks* e diálogos;
f) considerar as regras básicas para a formatação de telas: no tocante a aparência, formato e conteúdo das telas, utilização da cor, janelas, ícones e gráficos.

5.3.2 Paradigma Lições-Construtor-Analisador

O modelo proposto está fundamentado na combinação dos elementos das teorias pedagógicas comportamentalista e construtivista. Inicialmente projetado e experimentado na construção do software ASA (Animação e Simulação de Algoritmos) para o ensino de algoritmos no SENAC; estruturado em três módulos: Lições, Construtor e Analisador (GUIMARÃES, 2001).

O módulo Lições, semelhante às lições no livro-texto, ou uma aula expositiva ou uma fita de vídeo, é o primeiro contato do aluno, apresentado com imagens e texto. As animações são divididas em duas categorias: Conceito e Simulação. Conceitos são representações concretas de algo abstrato a ser ensinado. Por exemplo: representar a memória do computador (abstrato) através de gavetas (concreto). Simulações correspondem à representação mais fiel possível do mundo real. Tanto nos conceitos como nas animações, o aluno interage com o sistema, controlando a velocidade e o

número de vezes que cada animação será executada, fornecendo dados de entrada, quando solicitado. Por meio das animações, o aluno recebe *feedback* implícito; e, por meio de exercícios, o *feedback* explícito em qualquer tipo de mídia (texto, som, imagem, vídeo) e ainda registra a resposta na ficha do aluno.

No módulo Construtor, o aluno aplica o que foi apresentado no módulo Lições. O aluno cria e executa a simulação, podendo ser criada mais de uma forma de representação. O sistema também envia o *feedback* ao aluno e registra seu trabalho na Ficha do Aluno.

No Módulo Analisador, o instrutor pode analisar os resultados da interação do aluno através de um *log* (para um arquivo texto), de todas as atividades do aluno. O instrutor pode analisar, além dos resultados, os passos intermediários, constituindo uma forma de acompanhar o desenvolvimento dos alunos. Uma forma de aperfeiçoamento pode ser trabalhar esse log, classificando os resultados, gerando estatísticas e, baseado nisso, interagir mais efetivamente com o aluno.

Segundo o autor, o paradigma descrito permite que os desenvolvedores, independente do conteúdo temático da aplicação, sua natureza ou grau de dificuldade, lancem mão de recursos de visualização, operem com volumes expressivos de exercícios (o que implica o uso de simulações gráficas e requer participação ativa e intensiva dos usuários na sua resolução) e empreguem meios para rastrear o processo de aprendizagem e construção do conhecimento, inclusive com a possibilidade de gerar e fornecer registros estatísticos.

5.3.3 Design Patterns

Este modelo foi apresentado no Simpósio Brasileiro de Informática na Educação, com o objetivo de uso, por um público formado por pessoas que planejam, executam e avaliam projetos de informática educativa (CAMPOS, et al, 1999).

O padrão proposto nesse modelo adota uma abordagem menos estruturada e mais conceitual, tratando de gestão de projetos com documentação de modelos e heurísticas para confecção, revisão e validação dos projetos. O padrão é centrado na instituição de ensino como um núcleo formado por alunos, professores e todos os problemas relacionados aos projetos de utilização do computador na educação. É uma proposta que permite relacionamentos inter e intrapadrões, garantindo dinâmica na

utilização do modelo (CAMPOS, et al, 1999).

Os padrões são divididos em quatro grandes grupos: o primeiro trata de padrões de tecnologia de hardware e software; o segundo trata de desenvolvedores de software educacional; o terceiro está relacionado à política externa e, o quarto à instituição escolar.

O segundo grupo busca soluções para questões como:

- Qual o design instrucional para desenvolvimento de software baseado no behaviorismo?
- Qual o design instrucional para desenvolvimento de software baseado na teoria de Gagné?
- Qual o modelo para desenvolvimento de software baseado no construtivismo?
- Qual o modelo de design instrucional para o enfoque behaviorista, utilizando as tecnologias de rede e hipermídia?
- Qual o modelo para o enfoque construtivista, utilizando as tecnologias de rede e hipermídia?
- Quais as características do ambiente educacional behaviorista?
- Quais as características do ambiente educacional construtivista?
- Quais as características do software educacional que atendem ao novo paradigma educacional?
- Como vender e divulgar software educacional?
- O que diferencia o ciclo de vida de um software educacional?
- Como validar um software educacional?
- Como avaliar um software educacional?

O estudo dessas questões gerou um padrão proposto para o ciclo de vida do software educacional e a definição das etapas fundamentais para o processo:

a) definição do problema: O que diferencia o ciclo de vida de um software educacional? Quais são as etapas fundamentais?

b) Forças:

- O método tradicional de desenvolvimento de software, no qual os programadores sozinhos completavam o projeto foi ultrapassado;
- Métodos, procedimentos e ferramentas aumentam a produtividade e a

qualidade dos produtos;

- Projetos de desenvolvimento de software educacional, além de envolverem em seu desenvolvimento uma equipe multidisciplinar, devem refletir os objetivos educacionais propostos e o ambiente de aprendizagem almejado, criando situações que estimulem o desenvolvimento das habilidades desejadas;
- O modelo de ensino/aprendizagem selecionado no software, isto é, a filosofia de aprendizagem subjacente ao software é uma etapa a ser específica “a priori”, no desenvolvimento do software educacional e que vai determinar o seu desenvolvimento;
- A discussão pedagógica que precede a análise de requisitos é determinante no desenvolvimento do software e na determinação do seu ciclo de vida.

c) solução: adotar um processo de desenvolvimento de software educacional composto do modelo de ciclo de vida de prototipagem evolutiva, acrescido da etapa inicial da identificação do ambiente educacional das características do público-alvo e da definição dos objetivos para a aprendizagem e avaliação por parte de professores e alunos, para que novos requisitos sejam incorporados.

Para um padrão de desenvolvimento de um ambiente educacional construtivista foi proposto:

a) problema: quais as características do ambiente educacional baseado no construtivismo?

b) Forças:

- A ênfase de Piaget nas interações do sujeito com o objeto, como pré-requisito para a internalização de operações cognitivas, estímulo à manipulação de material concreto nas primeiras séries. Sua descrição do desenvolvimento por meio da auto-regulação evidenciou a necessidade de atividades do aluno e resolução de problemas;
- No modelo de Bruner, os processos internos têm muita importância e o produto final tem importância secundária;
- Ausubel argumenta ser necessária a explicitação e a evidenciação da

aprendizagem, denotando uma aprendizagem significativa, por meio de subsensores;

- Os construtivistas privilegiam os metaobjetivos e as estratégias internas para a produção do conhecimento. É a pedagogia de projetos;
- Nos ambientes de aprendizagem construtivistas, os alunos possuem muito mais responsabilidades sobre o gerenciamento de suas tarefas e seu papel no processo é de colaborador ativo;
- A ênfase é centrada no pensamento crítico;
- A progressão da aprendizagem é não-linear;
- O processo educacional é centrado no aluno;
- A interação se faz com o mundo real.

c) solução: definir os macroobjetivos e os contextos, para incentivar a construção do conhecimento e incentivar a participação do aluno no processo. A avaliação é qualitativa. Considerar a não-linearidade, a escolha de caminhos navegacionais por parte do estudante e a liberdade na busca da informação. Propor problemas realistas, interessantes e relevantes aos alunos e permitir testar diversas soluções. Estimular a colaboração, o diálogo e a negociação no trabalho em grupo, como forma de encorajar múltiplas interpretações.

O ambiente educacional construtivista deve seguir algumas heurísticas que irão contribuir para a construção do conhecimento:

- propor ambientes que permitam produção e conhecimento e a compreensão sob múltiplas perspectivas;
- propor problemas contextualizados e compatíveis com o conhecimento externo à sala de aula;
- permitir interpretação significativa e reflexiva;
- incentivar o pensamento crítico;
- encorajar a troca de idéias e testagem das alternativas;
- fornecer assistência ao aluno, ao contexto da aprendizagem e ao processo.

Os autores observam a importância do estabelecimento de padrões, como uma forma de disseminação de soluções obtidas através de muitas práticas, reduzindo a

complexidade da realidade e podendo fornecer ações que levem à solução de problemas. Ressalta a necessidade da utilização desses padrões, no entendimento, modificação e adaptação às necessidades específicas do usuário, para que se torne uma atividade colaborativa e em que a experiência de um venha a contribuir para o aperfeiçoamento e ampliação deste sistema.

5.3.4 Método da ULBRA

Este modelo (SILVEIRA, 2001) descreve o desenvolvimento de software educacional em forma de *slides* de apresentação. Para o autor, o desenvolvimento de softwares educacionais com qualidade é necessário à observação de algumas etapas, desde a concepção até a implantação dos mesmos:

1) definição do ambiente de aprendizagem: seguindo a filosofia de aprendizagem e o modelo de ensino-aprendizagem;
2) análise da viabilidade: definição de hardware, o cálculo dos custos e cronogramas;
3) seleção do tipo de documento: documentos que podem ser utilizados por diversos usuários (base de conhecimento sólida e consistente) ou produtos que não têm compromisso didático-pedagógico (exploratório);
4) seleção do método de autoria:
   - linguagens de programação: necessidade de incluir um programador na equipe de desenvolvimento;
   - linguagens de autoria: possuem rotinas específicas para o desenvolvimento de softwares em multimídia, com recursos como editor de texto, editor gráfico, inserção de sons e imagens, entre outros. São exemplos os softwares Multimedia Toolbook, Macromedia Director, Visual Class e Everest;
   - sistemas de autoria: são ambientes em que o usuário pode criar um software multimídia, sem ter que dispor de um conhecimento aprofundado de programação. Com um conhecimento básico é possível desenvolver softwares, tais como: apostilas eletrônicas, exercícios de fixação e tutoriais.
5) planejamento da interface:

6) planejamento do documento:
7) seleção do sistema de autoria e das ferramentas,

5.3.5 Modelo de Ciclo de Vida

O modelo foi publicado e disponibilizado na Internet, desenvolvido pela professora Neide Santos, traça uma visão geral sobre as teorias de aprendizagem que embasam esse modelo e noções básicas sobre software educacional, culminando na descrição do modelo proposto (SANTOS, 2000).

A proposta segue o modelo do ciclo de vida abordado pela Engenharia de Software clássica compreendendo as seguintes fases:

a) análise: fase em que se analisa qual é a melhor solução para o problema, definindo o ambiente educacional inclusive;
b) projeto: fase em que se define o plano de desenvolvimento do software (especificação, design, diretrizes de interface). Nesta fase, é sempre conveniente usar algum modelo/método para suporte da modelagem;
c) codificação: fase da escolha da plataforma de hardware e software em que o software será implementado e a própria implementação;
d) avaliação: definição dos critérios de avaliação e marcos de avaliação do processo de desenvolvimento e do produto;
e) manutenção: implantação e controle das versões do software.

A proposta está baseada nesse modelo de ES, observando as características das teorias de aprendizagem e o formato que irá assumir o software. Independente do enfoque adotado, o desenvolvimento de software segue as etapas não necessariamente seqüenciais:

a) etapa 1: definição do tema a ser abordado no software;
b) etapa 2: identificação dos objetivos educacionais da aplicação e do público-alvo;
c) etapa 3: definição do ambiente de aprendizagem;
d) etapa 4: modelagem da aplicação;
e) etapa 5: planejamento da interface;
f) etapa 6: seleção de plataforma de hardware e software;
g) etapa 7: implementação;

h) etapa 8: avaliação;

i) etapa 9: validação.

As etapas 1, 2 e 3 correspondem à fase da análise do ciclo de vida clássico. Nessa fase, define-se a concepção pedagógica e o embasamento teórico de aprendizagem, adequado ao tipo de software selecionado.

O ponto crítico dessa proposta está na definição do ambiente de aprendizagem e na modelagem da aplicação. Nessa etapa, deve-se garantir a definição do grau de interatividade do usuário com o software, o atingimento dos objetivos educacionais e sua adequação ao público-alvo e o respeito às características do ambiente de aprendizagem escolhido.

É aconselhável o estabelecimento de um padrão de desenvolvimento baseado nas etapas acima descritas. Conforme a autora, existem três tipos de padrões de software:

a) padrões conceituais: são padrões em que as formas são descritas por meio de termos e conceitos de um domínio de aplicação, são baseados em metáforas e restritos a um domínio de aplicação;

b) padrões de projeto: são padrões cujas formas são descritas por meio de construções de projetos de software, por exemplo, objetos, classes, herança, agregação e uso-relacionamento. Padrões de projeto complementam, ou são construídos sobre padrões conceituais;

c) padrões de programação: são padrões cujas formas são descritas por meio de construções de linguagens de programação. Padrões de programação implementam padrões de projeto, utilizando uma linguagem de programação específica.

Os padrões disponibilizam para muitos um conhecimento adquirido, validado e medido, porém, não especificam uma solução detalhada. Ajudam na redução da complexidade de muitas situações da vida real.

5.4 CONCLUSÃO DO CAPÍTULO

Segundo Campos (2001), para avaliar um software educacional, temos que considerar além das características inerentes ao produto, os atributos de domínio e as tecnologias específicas. Sabe-se que as teorias de aprendizagem refletem visões

profundamente diferentes sobre como ocorre o aprendizado e essas visões têm impacto nos modelos de software educacional. Acredita-se que a avaliação começa pela identificação do seu ambiente, ou seu potencial uso para um determinado ambiente educacional.

Steen Larsen, psicólogo e professor da Royal Danes School of Educational Studies na Dinamarca, em entrevista concedida à Revista Byte Brasil, em outubro de 1996, acha que não é necessário criar softwares para a educação, não é essencial. Pode-se usar a tecnologia de duas maneiras: como máquinas de treinamento ou para resolver problemas, que é a melhor alternativa. Para isso tem-se os softwares aplicativos; do tipo processador de texto, planilha de cálculo, desenhos gráficos, de apresentação multimídia. Os computadores são, antes de tudo, ferramentas de trabalho, são um meio e não finalidade. No entanto, nada impede de criar aplicativos voltados para educação (LARSEN, 1996).

Nos modelos estudados, observam-se a preocupação da atuação de equipes multidisciplinares, tanto na avaliação como no desenvolvimento de um software educacional, independente de que classificação irá receber o produto final. O contexto de seu uso determina a qualidade de um software educacional. Parece claro que a estrutura de comunicação entre o software e o usuário é que determina a qualidade de seu uso, quanto mais rico em recursos for o ambiente e maior o grau de interatividade, maior será o grau de envolvimento do usuário.

Neste capítulo procurou-se, sobretudo, identificar e tipificar as características próprias dos modelos de avaliação e desenvolvimento de software educacional, para que possam ser qualificadas como suporte para a aprendizagem e recurso pedagógico do professor.

6 PROPOSTA DE METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE EDUCACIONAL

Essa proposta foi elaborada a partir da revisão dos vários modelos de avaliação de produto e processo de software, baseada nas ferramentas de Engenharia de Software e nas normas ISO, sobre o processo e produto de software. Apoiou-se também no estudo ergonômico de interfaces, como o TICESE e a norma ISO 9241, nas propostas filosóficas e pedagógicas de conhecimento e aprendizagem que fundamentam os modelos já propostos, de desenvolvimento de software educacional. Todas as propostas estudadas e analisadas apresentam semelhanças quando se trata da questão de ambiente educacional, tendo uma filosofia a seguir, um modelo de processo ensino-aprendizagem, mas não esclarece um paradigma computacional para modelagem e desenvolvimento da aplicação. O que se pretende com a presente proposta é estabelecer critérios para o desenvolvimento de software para uso educacional integrando as filosofias educacionais com um modelo de desenvolvimento baseado em normas e critérios ditados pelas normas internacionais, amplamente aceitas no universo de construção de softwares.

Não se pode negar a contribuição da informática na resolução de problemas de ensino, principalmente sua interferência na democratização do ensino. Essa mudança implica uma revolução do perfil do profissional educador, para enfrentar o desafio de ensino e aprendizagem. Galvis (1999) diz que é um processo vital e permanente, que vai além da escolaridade, envolvendo a vida familiar, escolar, de lazer e de trabalho. Essas entidades vão-se transformando, de simples consumidores de soluções informatizadas, em parceiros estratégicos no desenvolvimento de habilidades, na expansão de suas capacidades e em sua visão de mundo, de uma perspectiva própria.

O uso de processos educativos informatizados pode-se dizer que depende desde a construção de softwares, como um pacote pronto, até o desenvolvimento de ambientes educativos interativos de ensino e aprendizagem. Galvis (1999) observa que um educador pode criar poderosos ambientes educativos, até mesmo com ferramentas de produtividade, como o Office (pacote da Microsoft); pode também usar a Internet como uma janela para o mundo globalizado, formando a consciência sobre os desafios da sociedade internacional, adequando ao nível do mundo real, mas não está praticando

Engenharia de Software Educacional, está apenas enriquecendo a educação com a informática.

A proposta objetiva a inclusão das normas de avaliação e desenvolvimento de softwares, internacionalmente aceitas e consolidadas como as normas ISO e CMM, onde podem ser usadas e quais os critérios a serem adotadas no seu uso.

6.1 DESCRIÇÃO DA PROPOSTA

Em qualquer desenvolvimento, devemos definir o objetivo que se quer alcançar com o produto final. A qualidade neste contexto não se limita ao cumprimento dos padrões associados ao tipo de software, inclui-se a observação dos critérios de pertinência, relevância e o uso que se quer fazer do computador, assim o desenvolvimento deve incorporar as características do ambiente educacional ao qual é direcionado.

A proposta está estruturada em etapas que podem ser implementadas em seqüência ou não, e adaptadas ao contexto de aplicação do software educacional.

Todo o embasamento estrutural da proposta baseou-se em modelo já consolidado, resultante de experiência prática de uma empresa de software, de porte pequeno, atuante na área administrativo-financeira de empresas industriais e comerciais, e na pública, como empresa terceirizada.

Resultado de esforços desde 1995, atualmente na terceira versão, com uma pessoa especializada em qualidade, contando com os esforços adicionais da área de desenvolvimento da organização.

Foram usadas para a definição da metodologia, uma série de referências conceituais, baseada em normas e métodos de trabalho, dentre eles:

- norma ISO/IEC 12207
- norma NBR 13596 (ISO/IEC 9126)
- norma NBR 9000-3
- modelo SPICE
- modelo CMM
- as propostas GQM (Goal Question Metrics)

Dentro das propostas, o GQM fornece um roteiro para a análise e definição de um plano de medição para o software, direcionando o trabalho proposto, fornecendo o

roteiro para a execução do Plano de Melhoria. Apresenta-se, também, como uma forma de generalizar, consolidar e disseminar as experiências por toda a empresa. O conjunto dessas técnicas dá à empresa uma forma de melhorar o processo de desenvolvimento de software baseado no reuso de experiências bem sucedidas.

O Modelo de Desenvolvimento de Software-MDS está estruturado em forma de roteiros dentro dos padrões que embasam o modelo.

No processo fundamental, estão os roteiros de: projeto preliminar, roteiro de atividade contínua, roteiros de desenvolvimento e guia de especificação de requisitos. No processo de apoio foram definidos os roteiros de revisão, documentação, avaliação do produto e gerência de configuração.

No contexto organizacional, estão os roteiros de acompanhamento, estimativas de software (pontos por função ou linhas de código) e roteiro de melhoria, além do processo de gerência.

6.1.1 Etapa Um – Caracterização do Projeto de Software

Na primeira etapa, definem-se as características de identificação dos objetivos do projeto de software, como o tema, os objetivos educacionais, a relação custo-benefício e a definição do público alvo.

Nesta etapa, é primordial que a equipe multidisciplinar já esteja composta, ou, na impossibilidade, os agentes interessados estejam definidos, com metas definidas e identificadas às diretivas do porquê, para quem e por quem, complementados posteriormente com o como e quando.

Os critérios a serem observados são definidos no ROTEIRO DE PROJETO PRELIMINAR (conforme apresentado na figura). O planejamento envolve a definição de prazos, organização da equipe, forma de trabalho, técnicas e ferramentas, fases, produtos, forma de revisão, aprovação e acompanhamento. A formação da equipe e a sua atuação influencia diretamente o andamento do projeto, nas questões de distribuição de tarefas, definição de papéis e gerenciamento de conflitos. Devem ser adotados indicadores (medidores) de nível de planejamento, nível de atuação da equipe, nível de execução de tarefas, adotando algum mecanismo flexível, observando os conflitos inerentes a esta etapa.

Devem ser estabelecidas as políticas, regras e diretrizes que reduzirão as opções de projeto, dentro de uma visão de longo prazo. Devem ser observados os requisitos funcionais e os não funcionais, a escala (tamanho) e a complexidade do sistema, bem como a necessidade de evolução ao longo do ciclo de vida, as restrições de orçamento, a capacitação da equipe e o equilíbrio entre a forma e a função do projeto de sistema (LANZUOLO, 2000).

6.1.2 Etapa Dois – Definição do Ambiente e Adequação à Educação

Na segunda etapa, define-se o ambiente de aprendizagem e os aspectos pedagógicos da aplicação, fundamentado nas características definidas na primeira etapa.

Define-se aqui a teoria de ensino-aprendizagem para o objetivo pretendido, o nível de aprofundamento do conteúdo, definição do perfil do público-alvo, o objetivo da aprendizagem, o contexto curricular e as formas e modalidades de avaliação da aprendizagem. Os ambientes devem refletir os diferentes contextos tempo/espaço (síncronos, assíncronos, distribuídos ou não, cooperativos) em forma de equipe, buscando a solução de problemas apoiados em algum conceito de estratégico de resolução de problemas.

Na construção conjunta das soluções, devem ser considerados e explorados: a dinâmica do processo, o envolvimento, a participação e a produção coletiva ou

individual, a atuação de cada um na dimensão de trocas e de cooperação e exploração de recursos, bem como a criatividade, a forma de representação das várias formas de informação (SANTOS, 2000).

Deve-se definir as estratégias cognitivas a serem adotadas, como o *group awareness*, *floor passing*, *scene analysis*, *brainstorm* e uma tecnologia intelectual que conduza a uma hipertecnologia, capaz de gerar um meta-ambiente de aprendizagem cooperativa (SANTOS, 2000).

O resultado desta etapa deve contemplar o design instrucional, para o desenvolvimento de software baseado em uma teoria de aprendizagem (construtivista, behaviorista, sócio-interacionista) e o que essas escolhas influenciarão no uso de tecnologias de hardware (de rede, hipermídia e outros).

6.1.3 Etapa Três – Definição dos Requisitos

Na terceira etapa, a aplicação da norma ISO 12207, sobretudo nos processos fundamentais dentro das características de desenvolvimento, com a definição dos atributos: requisitos do sistema, análise e projeto, análise dos requisitos do software e projeto do software, e no processo de apoio às características de documentação, garantia da qualidade, verificação e resolução de problemas.

São critérios explicitados no ROTEIRO DE DESENVOLVIMENTO e ROTEIRO DE ATIVIDADE CONTÍNUA e principalmente no GUIA DE ESPECIFICAÇÃO DE REQUISITOS. O guia deve contemplar a elaboração de um documento que seja a base de relacionamento desenvolvedor/usuário, agregar os aspectos que o software deve atender, ser independente de tecnologia, ser de fácil entendimento tanto pelos técnicos em informática como pelo usuário, apresentar uma estrutura que facilite o gerenciamento de requisitos durante o ciclo de desenvolvimento do produto, estar em conformidade com padrões internacionais que tratam do assunto e ser facilmente integrável a uma metodologia já existente. Na elaboração do escopo do guia foram utilizados os conceitos descritos na área-chave do CMM-nível 2: Gerenciamento de Requisitos e da Norma ISO/IEC 12207 – Processos de Ciclo de Vida de Software. Também referenciadas a NBR 13596 para Avaliação de Produto de Software – Características de Qualidade e Diretrizes para o seu uso.

A atividade de análise de requisitos tem como objetivo alocar os requisitos

identificados ao produto a ser desenvolvido. Essa análise pressupõe uma análise de riscos e de custos para a contemplação de um determinado requisito no software. Um produto comum da análise é o surgimento de novos requisitos, a identificação de dependência ou incompatibilidade de atendimento de requisitos ao mesmo tempo. Outra grande questão é que os requisitos mudam ao longo do tempo e, muitas vezes, requisitos que não foram alocados inicialmente, poderão ser realocados ao software conflitando com outros já alocados. Outro fator que contribui para a necessidade de uma análise criteriosa e um gerenciamento é que muitas vezes, nem todos os requisitos são identificados durante a atividade de especificação de requisitos, somente ficando clara a sua necessidade, quando o software já está sendo desenvolvido, necessitando de um gerenciamento organizado, tanto da inclusão como de deleção de requisitos ao produto de software, pois modificações, quase sempre, implicam alteração de prazos, de recursos humanos e de máquina, de alteração de tecnologia ou de ambiente de desenvolvimento.

6.1.4 Etapa Quatro – Definição da Interface

Na quarta etapa, seleção da interface a ser utilizada, tendo como base os requisitos definidos na etapa anterior, fortemente embasada nos princípios ditados na norma ISO 9241, com apoio nos critérios de ergonomia desenvolvido no TICESE (GAMEZ, 1998), a seleção das interfaces apoiou-se nos estudos da professora Neide Santos, intitulado “Design de Interfaces de Software Educacional”, com complementação no trabalho desenvolvido pela Dra. Venétia Santos e da Dra. Maria Cristina Zamberlan, publicado pela Fundação Mapfré do Brasil, intitulado “Projeto Ergonômico de Salas de Controle”, com referência ao capítulo que trata de “Configuração das Telas”, estudo envolvendo divisão, diagramação, diálogo, linguagem, cores, tamanho, brilho, fundo da tela, estímulos, fadigas e conseqüências. Também a professora Renata Vieira, da Unisinos, no trabalho “Projeto de Interfaces”, parte integrante do conteúdo da disciplina de projetos de interface.

6.1.5 Etapa Cinco – Implementação do Modelo

A etapa 5 caracteriza-se pela implementação do modelo, consolidando todas as etapas anteriores. Novamente faz-se uso da norma ISO 12207, com os atributos de

construção de software, integração do software, teste de software, manutenção e operação, com uso total dos processos de apoio, envolvendo as características de processos organizacionais, com a observação dos atributos de melhoria e treinamento. Complementa-se com a característica de “Reuso”, desde códigos até soluções e partes de projeto de software, descritas na futura norma ISO 15504.

Conforme Pressman (1995), a fase de testes ocupa, normalmente, 40% do tempo planejado para um projeto e a detecção de erro nessa fase provoca um aumento de 60% nos custos do projeto, justificando daí, a importância de um bom planejamento e controle de execução de testes.

6.1.6 Etapa Seis – Avaliação do Modelo

A etapa 6 envolve os trabalhos de avaliação de produto e processo, utilizando-se das normas ISO 9126, ISO 12207, ISO 15504, para as características técnicas de construção; ISO 9241 e critérios ergonômicos TICESE, para características de interface ergonômica de software e dos fatores educacionais, dentro das propostas didático-pedagógicas e epistemológicas.

6.1.7 Etapa Sete – Validação do Modelo

A etapa 7 envolve os trabalhos de validação da proposta. Pode-se fazer uso de métricas internas e externas e de qualidade em uso, no âmbito da norma ISO 9126, mas o mais importante é o envolvimento do usuário nesse processo.

Para Galvis (1999), o significado e a relevância de um software educacional tem a ver, em grande medida, com o grau de atenção que ele dedique ao campo vital do aprendiz, a seu ambiente psicológico, a suas experiências anteriores, suas expectativas, motivação interna, atitudes e aptidões. Concluindo, ele diz que alguns princípios podem ser úteis nesse processo: o uso de um enfoque estratégico, tratando o aprendiz como cliente; apoiar-se em marcos tecnológicos compartilhados, ou seja, a integração entre os responsáveis pela decisão (organização), responsáveis individuais (professores e estudantes) e os responsáveis pela tecnologia (toda a dinâmica tecnológica); e no descobrimento das expectativas, a partir de necessidades relevantes, isto é, a realização daquilo que apresenta valor ao cliente, dentro das identidades individuais e visão da organização, a cujo serviço esteja o ambiente educativo informatizado.

Nesta etapa reforçam-se os processos de apoio da norma ISO 12207, no que diz respeito à documentação, garantia da qualidade, validação, verificação, revisão conjunta, auditoria e resolução de problemas. Envolve os processos organizacionais com respeito à melhoria e treinamento; também nos processos fundamentais com os atributos de operação e manutenção principalmente.

A avaliação e validação deve ser realizada em todo o processo de ciclo de vida do software, podendo ser usados procedimentos e instrumentos, tais como: observações, experimentos, anotações das ocorrências, acontecimentos, questionários, entrevistas, simulação do funcionamento do processo, comparação com padrões já existentes, entre outros.

6.2 CONCLUSÃO DO CAPÍTULO

Recomenda-se, para implementação da proposta, um modelo de ciclo de vida de software, que se baseia na combinação de vários paradigmas, observando as potencialidades e fragilidades em cada paradigma, para atender às expectativas do cliente, maximizando o relacionamento desenvolvedor-usuário.

Pressmann (1995) salienta que o software evoluiu de uma ferramenta de análise de informações, para um meio de resolução de problemas especializados, modificando a indústria em si mesma. Por isso, tornou-se um fator limitante na evolução dos sistemas.

Pode utilizar-se de ferramentas que estabeleçam medidas para o estudo de viabilidade de projeto de software, como por exemplo, a técnica de Análise de Pontos por Função ou FPA-Function Point Analysis, desenvolvida pela IBM, no início da década de 70, a pedido de um grupo de usuários, com o objetivo de identificar as variáveis críticas que determinam a produtividade da programação.

Essa técnica avalia um software, medindo o valor das funções executadas pelos programas, em vez de códigos por linha. Em 1979, Allan Albrecht, prosseguindo as pesquisas, introduziu a técnica conhecida como Pontos por Função. Uma técnica baseada na visão externa do usuário, independente da linguagem utilizada, permitindo calcular o esforço de programação e auxiliando o usuário a melhorar o exame e avaliação de projetos. A última versão, publicada em janeiro de 1999 (V 4.1), é mais adequada para estimativas de software, mensurando dados como o custo, o tamanho,

performance, utilização de equipamentos, volume de transações, reutilização de código, o que chamou a atenção das organizações de desenvolvimento de software e da academia, sendo formado, em 1986, um grupo internacional de usuários da técnica, chamado International Function Point User Group-IFPUG, destinado a melhorar e disseminar informações da técnica.

Com as medidas embasadas em técnicas estáveis e consolidadas, torna-se amena a tarefa de estabelecer uma relação custo-benefício-tempo, descrita na etapa inicial.

É preciso reforçar que as tecnologias não mudam, necessariamente, a relação pedagógica e nem substituem o professor, mas modificam algumas das suas funções. O professor transforma-se em um estimulador da curiosidade do aluno, por querer conhecer, por pesquisar, por buscar uma informação mais relevante. E, num segundo momento, coordena o processo de apresentação dos resultados. O professor deve estimular o aluno a transformar a informação em conhecimento e conhecimento em saber, em vida, em sabedoria.

7 - CONCLUSÃO

Em reunião organizada pela Sociedade Softex, em Brasília, reunindo pesquisadores, dirigentes de instituições de base tecnológica e empresários do setor, para discutir o presente e futuro da indústria de software no Brasil, o então Ministro da Ciência e Tecnologia, Ronaldo Sardenberg, destacou que os investimentos em pesquisa e desenvolvimento no Brasil, no período de 1993 a 1999, totalizaram US$ 2,7 bilhões e que o crescimento da indústria de software no país atingiu 19% em 1999, movimentando cerca de US$ 3,2 bilhões. Não há comparativo fidedigno para estes dados, apenas uma nota em um artigo publicado na Revista Bate Byte da CELEPAR, relatando que a indústria mundial de desenvolvimento de software movimentou, em 1999, valores acima de 600 bilhões de dólares. Pelos números, o perfil do Brasil é incipiente, muito pouco significativo.

Em um ambiente de enfoque estratégico de gerência da Qualidade Total, são requeridos a busca contínua de melhoria, prioridade à qualidade, uso de engenharia de processos, e facilidade de manutenção. Esta busca de melhoria dos ambientes de desenvolvimento, não apenas do produto, mas também dos processos, tem levado as empresas a buscarem modelos de sucesso, para formularem as suas estratégias de priorização da qualidade.

O conhecimento de modelos e normas eleva o potencial produtivo das pessoas envolvidas em desenvolvimento de software e direciona o seu trabalho para a obtenção de instrumentos fundamentais para a melhoria destes trabalhos; determina uma conduta gerencial mais globalizada e menos individualizada, monitora os procedimentos operacionais e também dos recursos da empresa a partir de indicadores básicos definidos.

Todos os modelos propostos são iguais na essência, uns influenciam outros, e muitos derivaram dos mesmos estudos preliminares, e os mais recentemente divulgados foram influenciados pelos mais antigos. Todos apresentam *aspectos* positivos e *suas* limitações.

Dos modelos apresentados, a ISO 9000-3 é a mais difundida, no entanto, não aborda diretamente a melhoria contínua do processo, levando a uma interpretação errônea de que o objetivo principal de sua aplicação seja a certificação da empresa. A

ISO/IEC 9126 são normas especificamente para o produto de software. O Capability Maturity Model-CMM, um modelo embasado em softwares encomendados pelo Departamento de Defesa dos EUA, aborda a melhoria contínua do processo com muita ênfase; no entanto, a abordagem de diversidade de empresas é irrelevante, e a sua aplicação em empresas pequenas torna-se pouco atraente e ineficaz. O modelo SPICE flexibilizou outros modelos anteriores como o CMM, TRILLIUM e o Bootstrap (modelo da Comunidade Européia), no entanto, dificulta a sua aplicação em organizações de pequeno porte.

Todos, inegavelmente, apresentam princípios de qualidade e os dissecam, porém se percebe que não existe um modelo ideal para aplicação. A modelagem, automatização de processos e a padronização indicam o caminho para a melhoria contínua dos processos e do produto de software. Entretanto, cada organização deve aprofundar-se no estudo dos modelos, no interesse dos objetivos enunciados, que sejam coerentes com as propostas e aplicá-los da melhor forma possível, dentro de seus objetivos e estratégias empresariais.

A proposição da metodologia para o desenvolvimento de softwares educacionais, parte do princípio de que não basta apenas a adequação a um padrão globalizado e generalista, mas sim, deve-se adequar em conformidade com os requisitos básicos de cada clientela, de cada segmento consumidor, de cada público espectador. Deve-se observar o universo de aplicação de produto de software como instrumental básico no processo de ensino-aprendizagem, em que, interagem, quem ensina e quem aprende, formando uma interface de comunicação. Comunicação que nem sempre se torna completa, porque, ou o instrumento é inadequado para um dos lados ou para ambos, cortando a inter-relação. Conclui-se, dessa forma, que é muito importante a adoção de padrões de qualidade internacionais, é importante ter-se um processo-padrão de desenvolvimento de software, é importante basear-se nas filosofias e práticas educacionais, mas muito mais importante é traduzir a interface do usuário, o aprimoramento para uma funcionalidade específica para um segmento especializado.

Recomenda-se, para trabalhos futuros, os seguintes itens:

- Definição de atributos específicos para cada característica descrita nas etapas; atributos estes especializados para uso educacional;

- ❖ Validação da proposta, formalizando o ciclo completo das etapas;
- ❖ Adaptação da proposta para todos os níveis educacionais, isto é, uma proposta especializada para cada nível ou grau de escolaridade, e dentro destes, ainda, em faixa etária;
- ❖ Avaliação da proposta no contexto de qualidade em uso;
- ❖ Validar a proposta a luz dos princípios didático-pedagógicos da psicolingüística na sua contribuição para o processo ensino-aprendizagem;
- ❖ Disponibilizar a proposta para o Núcleo de Desenvolvimento de Software Educacional do Curso de Licenciatura em Informática da Faculdade da Cidade de União da Vitória-FACE;
- ❖ Aplicar a proposta ao desenvolvimento do software "Controle Financeiro de Empresas" para o Curso de Ciências Contábeis da FACE;
- ❖ Na continuidade desta linha de pesquisa, pretende-se desenvolver uma proposta específica para usuários portadores de deficiências.

E, por fim, utilizando das palavras de Ximenes (in Jornal Informasoft, 1995):

.... o problema critico das organizações modernas é fazer com que o conhecimento individual de seus profissionais se multiplique em conhecimento organizacional. Procuro ferramentas que me permitam registrar, ordenar e distribuir inteligentemente este conhecimento, no local de trabalho, na hora, no formato e ao nível de detalhe que for necessário. Just-in-time knowledge!

8 REFERÊNCIAS BIBLIOGRÁFICAS

ABNT-Associação Brasileira de Normas Técnicas. Normas Técnicas Brasileiras-Subcomitê de Software-SC-21:10, CE-21:101.01 – Qualidade de software. URL: http://www.pr.gov.br/celepar/abntsoftware/ce10101 (21/05/2000).

ABNT-Associação Brasileira de Normas Técnicas. ISO 9126/NBR 13596-Avaliação de Produto de Software-Características de Qualidade e Diretrizes para o seu Uso. 1996.

ABNT-Associação Brasileira de Normas Técnicas. NBR ISO 9000-3-Diretrizes para aplicação da BNR 19001 ao desenvolvimento, fornecimento e manutenção de software. 1993.

ABNT-Associação Brasileira de Normas Técnicas. NBR ISO 12207-Processo de Ciclo de Vida de Software.1998.

ABNT-Associação Brasileira de Normas Técnicas. NBR ISO/IEC 12119-Pacotes de Software-Teste e Requisitos de Qualidade. 1998.

ABNT-Associação Brasileira de Normas Técnicas. NBR ISO/IEC 14598-5-Avaliação de Software-Parte 5: Processo para avaliadores. Out/1999.

ABNT-Associação Brasileira de Normas Técnicas. Norma Mercosur NM-ISO 8402-Gestão da qualidade e garantia da qualidade-Terminologia. 1997.

ABNT- Grupo CE-21. Usabilidade de Software. Seminário de Qualidade de Software. Ago/2001. Curitiba-PR.

ABNT- Grupo CE-21. Modelo de Qualidade e Usabilidade de Software. Seminário de Qualidade de Software. Ago/2001. Curitiba-PR.

ABNT-Associação Brasileira de Normas Técnicas. NBR ISO 8402 - Gestão da qualidade e garantia da qualidade - Terminologia. Jul/1993.

ABNT-Associação Brasileira de Normas Técnicas - CB-21-SC-21:10 Subcomitê de Software. NBR ISO/IEC 9126 - Tecnologia de Informação-Avaliação de produto de Software - Características de qualidade e diretrizes para o seu uso.

ANAIS - XI Conferência Internacional de Tecnologia de Software: Qualidade de

Software. Curitiba: Centro Internacional de Tecnologia de Software, 2000. 190p.

ANAIS – SQM'97-Software Quality Management 5th Annual International Conference. Inglaterra: 1997.

ANAIS - VII Conferência Internacional de Tecnologia de Software: Qualidade de Software. Curitiba: Centro Internacional de Tecnologia de Software, 1996. 244p.

ANTONIONI, José A. e ROSA, Newton Braga. Qualidade em Software: Manual de aplicação da ISO-9OOO São Paulo: Makron Books, 1995. 108p.

APOSTILAS DO CURSO DE METODOLOGIA JACKSON. Curitiba: CELEPAR, 1980.

ARGOLLO Jr, Miguel. Portabilidade de Interfaces Gráficas. Revista Byte Brasil. Mai/1996.

ARGOLLO Jr, Miguel. Aplicativos com Interfaces Gráficas. Revista Byte Brasil. Jul/1996.

ASSESPRO-Associação das Empresas de Processamento de Dados. Softwares Educacionais: a um passo da definição das regras para a Rede Pública. Nº 206. mar/abr/1999. URL: www.assespro.com.br/direto.htm. (30.07.2001).

BANAS QUALIDADE. Guia de Softwares para Qualidade. Revista, nº 111, ano X. ago/2001.

CAMPOS, F. et all. Dez Etapas para o Desenvolvimento de Software Educacional do Tipo Hipermídia. Barranquilla: Uninorte, III Congresso Ibero-Americano de Informática Educativa, 1996.

CAMPOS, Fernanda C.A., CAMPOS, Gilda H. B. e ROCHA, Ana Regina C. da. Design Patterns: Um Sistema para Gestão de Informática Educativa. Simpósio Brasileiro de Informática na Educação, 1999.

CAMPOS, Fernanda & GAIO, Fátima. Perfil do Mercado Brasileiro de Software Educacional: Um Estudo Exploratório. UFJF-Universidade Federal de Juiz de Fora, Departamento de Ciências da Computação.

CAMPOS, Gilda Helena B. A Qualidade em Software Educacional. URL: www.casadaciência.com.br/artigos. (30.07.2001).

CARAM, Carlos Alberto. Guia De Softwares Para A Qualidade. Revista Banas

Qualidade. Ago/2001. nº 111. ano X.

CHAVES, Eduardo C. O Computador como Tecnologia Educacional. URL: www.edutecnet.com.br/artigos. (30.06.2001).

CITS-Centro Internacional de Tecnologia de Software. Jornal, ano 5, nº 08, out/2000.

CTI-Centro Tecnológico para Informática. Pesquisa sobre Gerência de Configuração de Software/1999. Instituto de Computação-PQPS-Programa de Qualidade e Produtividade em Software. URL: http://www.ic.cti.br/ic/pqps/gcs (19/06/2000).

CTI-ITI-Instituto Nacional de Tecnologia da Informação. Método ITI-Avaliação de Pacotes de Software. Seminário de Qualidade de Software. Ago/2001. Curitiba-PR.

CYBIS, Walter de A. Fundamentos da Abordagem Ergonômica para IHC. Anexos 1 – M.A.D. e Anexo 2-Ergolist 1997. URL: http://www.inf.ufsc.br (20/05/2000).

CYSNEIROS, Paulo G. Professores e Máquinas: Uma Concepção de Informática na Educação. URL: www.edutecnet.com.br/textos/alia/PROINFO. (13.02.2001).

CORDEIRO, Marco Aurélio. O Panorama Atual da Indústria de Software. Revista Bate Byte. Nº 97. jun/2000.

COSTA, Fernando A. Contributos para um Modelo de Avaliação de Produtos Multimedia Centrado na Participação de Professores. Lisboa, 1999.

ESI-European Software Institute. Presents Status of the ISO/IEC 15504 Standard. ESI Articles and Publications. URL: http://www.esi.es/Publications/Articles (24/06/2000).

FABRIS, Adilson. Orientações básicas para construção de interfaces. Revista Bate Byte 90. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb90 (07/06/2000).

FERNANDES, Aguinaldo Aragon. De. Métricas-Garantindo a qualidade do projeto, processo e produto. São Paulo: Atlas, 1995. 421p.

FERREIRA, Márcia O. V., GUGLIANO, Alfredo Alejandro e colaboradores. Fragmentos da Globalização na Educação-uma perspectiva comparada. Porto Alegre: Artes Médicas Sul, 2000. 263p.

FREIRE, Paulo. Pedagogia do Oprimido. Rio de Janeiro: Paz e Terra, 1987.

GALVIS, Alvaro H. & PANQUEVA, D. Ed. Software Educativo Multimídia-Aspectos Críticos no Seu Ciclo de Vida. Santafé de Bogotá: UNIANDES, 1999.

GAMEZ, Luciano. TICESE-Técnicas de Inspeção de Conformidade Ergonómica de

Software Educacional. Dissertação de Mestrado. Portugal: Universidade do Minho, 1998.

GIL, Antonio de Loureiro. Qualidade Total em Informática. São Paulo: Atlas, 1992. 144p.

GIRAFFA, Lucia M. M. & VICCARI, Rosa M. Estratégias de Ensino em Sistemas Tutores Inteligentes Modelados Através da Tecnologia de Agentes. VIII SBIE.

GUIMARÃES, Mário André M. Um Paradigma para o Desenvolvimento de Software Educacional. São Paulo: Botem Técnico do SENAC, 2001.

HEEMANN, Vivian. Avaliação Ergonômica de Interfaces de Base de Dados por Meio de Checklist Especializado. Dissertação (pós-graduação em Engenharia de Produção da UFSC). 1997. URL: <http://www.inf.ufsc.br/disserta97/heemann (05/1999).

ISO-International Organization for Standardization. 9000 + 14000, ISO 9000 revisions - Draft Internacional Standards. ISO NEWS, vol 8, nº 3, mai/jun/1999. URL: http:www.iso.

JEFFRIES, Robin, et. Alii. User Interface Evaluation in The Real World: a Comparison of Four Techniques. In CHI-91 Human Factors in Computing Systems, 1991.

LABIUTIL - Laboratório de Utilizabilidade. URL: <http://www.labitutil.inf.ufsc.br (20/05/2000).

LANZUOLO, Riccardo. Arquitetura de Software. Boletim. MSA-INFOR. nº 26. 2000.

LARSEN, Steen. Educação multimídia. Entrevista. Revista Byte Brasil, out/1996.

LÉVY, Pierre. As tecnologias da Inteligência – O Futuro do Pensamento na era da Informática. Rio de janeiro: Trans, 1993.

MACHADO, Cristina A. F., REINEHR, Sheila e JAEGGER NETO, José I. Desenvolvimento de Software de Acordo com Padrões Internacionais de Qualidade. Minicurso 2 XI CITS. Curitiba: CITS, 2000.

MACHADO, Cristina A. F., OLIVEIRA, Luiz C. de, FERNANDES, Rosane A. Inserção de Qualidade no Processo Metodológico. Revista Bate Byte 47. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb47 (08/06/00).

MACHADO, Cristina A. F. Base Conceitual para a Metodologia de Desenvolvimento de Software da CELEPAR. Revista Bate Byte 57. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb57 (08/06/00).

MACE, Eduardo. A Dura Vida do Software Educativo no Brasil. Revista Super Interessante, abr/2001.

MAFFEO, Bruno. Engenharia de Software e Especificação de Sistemas. Rio de Janeiro: Campus, 1992. 484p.

MANUAL BANAS. Revisão das normas série ISO 9000. Separata da Revista Controle da Qualidade nº 25.

MARTIN, James. Engenharia da Informação: Introdução. Tradução: Follow-up Traduções e Assessoria de Informática. Rio de Janeiro: Campus, 1991. 196p.

MARTINS, Vidal. O Processo Unificado de Desenvolvimento de Software. Revista Bate Byte 89. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb89 (08/06/00).

MCT-Ministério da Ciência e Tecnologia. SEPIN-Secretaria de Política de Informática e Automação. Relatório Bienal da pesquisa "Qualidade no setor de software Brasileiro" de 1997. Brasília: MCT, 1998. Nº 02.

MCT-Ministério da Ciência e Tecnologia. SEPIN-Secretaria de Política de Informática e Automação. Relatório Bienal da pesquisa "Qualidade no setor de software Brasileiro" de 1999. Brasília: MCT, 2000. Nº 03.

NEVES, Luciane. A atividade de teste durante o ciclo de vida do software.Revista Bate Byte 88. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb88 (08/06/00).

OLIVEIRA, André Luis C. de. Desenvolvimento acelerado e evolução tecnológica natural. Revista da ASSESPRO. Ano 1, nº 3. Mai/2000.

OLIVEIRA, Luiz Carlos de A., SILVA, Lilia Pinheiro C. Modelo CMM-uma visão geral. Revista Bate Byte 88. <http: //www.pr.gov.br/celepar/celepar/batebyte/bb88> (21/05/2000).

PAULK, Mark C., CURTIS, Bill, CHRISSIS, Mary Beth and WEBER, Charles V. Capability Maturity Model for Software. Version 1.1. Pittsburgh (Pennsylvania): Software Engineering Institute-Carnegie Mellon University, 1993.

PADILHA, Emannuel. O Significado da qualidade em educação. Revista Bate Byte 44. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb44 (08/06/00).

PLANETA EDUCAÇÃO. Consulte um especialista. URL: <http://www.planetaeducação.com.br> (30.06.2001).

PRESSMAN, Roger. Engenharia de Software. São Paulo: Makron Books, 1995.

ROCHA, Ana Regina C. da, MACHADO, Luis Filipe C. Uso de Normas e Modelos de Maturidade na Definição de Processos de Software. Minicurso 4-XI CITS. Curitiba: CITS, 2000.

SALVIANO, Clênio, et al. Experiência de Avaliação de Processos e Planejamento de Melhoria Utilizando a Futura Norma ISO/IEC 15504 (SPICE). WQS-99-Workshop de Qualidade de Software do SBES-99. Florianópolis, out/1999.

SANTOS, Neide. Desenvolvimento de Software Educacional. Notas de aula. 2000.

SANTOS, Neide. Design de Interfaces de Software Educacional. I-editora, 2001.

SANTOS, Solange C. Ambientes & Softwares de Apoio à Telemática. UFRGS-PGIE97. URL: <www.penta.ufrgs.br/~solange/artigo1> (03.07.2000).

SCALET, Danilo. Processo de Desenvolvimento de Sistemas com Qualidade. Revista Bate Byte 39. URL: http://www.pr.gov.br/celepar/celepar/batebyte/bb39 (08/06/00).

SILVEIRA, Andre Luis M. Critérios para Análise de Interfaces. UNISINOS: Dissertação de Mestrado. 1999.

SILVEIRA, Sidnei R. Qualidade de Software Educacional. ULBRA, apostila de aula, 2001. URL: www.ulbra.tche.br (31.07.2001).

STANDISH GROUP. Chaos University 2000. URL. www.standishgroup.com (30.06.2001).

TSUKUMO, Alfredo N. et al. Avaliação e Melhoria da Qualidade do Produto de Software Educacional "Escritor". Anais do X Simpósio Brasileiro de Engenharia de Software-SBES 96. São Carlos: out/1996.

VALENTE, José Armando. Computadores e conhecimento: repensando a educação. Campinas: Unicamp, 1998.

VIEIRA, Renata. Projeto de Interfaces. Unisinos: apostila de aula, 2000.

VIEIRA, Fábia M. S. Avaliação de Software Educativo: Reflexões para uma Análise Criteriosa. URL: www.edutecnet.com.br/Alia/MISC. (13.02.2001).

WEBER, Kival C. Qualidade e Produtividade em Software-Termo de referência do subprograma de QPS do Programa Brasileiro da Qualidade e Produtividade. Brasília:QA&T Consultores Associados, 1994.

WEINBERG, Gerald M. Software com Qualidade - Pensando e Idealizando Sistemas. Tradução: Flávio Deny Steffen. São Paulo: Makron Books, 1993. 387p.

WEINBERG, Gerald M. Software com Qualidade - Medidas de primeira ordem. V.2. Tradução: José Carlos Barbosa dos Santos. São Paulo: Makron Books, 1994. 450p.

XIMENES, Fernando Barcellos. Olhando para o futuro. Florianópolis, 1995. Jornal Informasoft, edição nº 70.

YOURDON, Edward. Administrando Técnicas Estruturadas-Estratégias para desenvolvimento de Software dos anos 90. Tradução:Daniel Vieira. Rio de Janeiro: Campus, 1988. 244p.

ZAMBERLAN, Maria C. & SANTOS, Venétia. Projeto Ergonômico de Salas de Controle. Fundação Mapfré do Brasil. URL. www.edutecnet.com.br/artigo/ergonomia (31.07.2001).