DEVELOPMENT OF METHODS FOR SPECIFIC CHECKLISTS TO EDUCATIONAL SOFTWARE

Max Robert Marinho [1], Toni Amorim de oliveira [2] , Claudinéia Pereira de Sousa [3]

[1]Professor Auxiliar do Departamento de Computação Campus Universitário de Alto Araguaia, Universidade do Estado de Mato Grosso (UNEMAT)
Rua: Santa Rita, n. 148, Centro, CEP: 78.780-000, Alto Araguaia – MT – Brasil

[2]Professor Assistente Mestre do Departamento de Computação Campus Universitário de Alto Araguaia, Universidade do Estado de Mato Grosso (UNEMAT)
Rua: Santa Rita, n. 148, Centro, CEP: 78.780-000, Alto Araguaia – MT – Brasil

[3] Academica do curso de Licenciatura em Computação, Campus Universitário de Alto Araguaia, Universidade do Estado de Mato Grosso (UNEMAT)
Rua: Santa Rita, n. 128, Centro, CEP: 78.780-000, Alto Araguaia – MT – Brasil

maxrobertm@gmail.com, toniamorim@gmail.com, neiafofa2@hotmail.com,

Abstract: It identifies a great difficulty in conceptualizing a SE (educational software), classify and establish its characteristics. For an effective assessment of an SE is required to draw up specific criteria for evaluation of each modality. The quality of an SE is widely discussed among educators and experts who need to learn how to evaluate software on the market. This work studies certain categories of software (training, application, tutorial and games) as to its definition, and elaboration of specific checklists for each modality, which consists of questions asked of the software to check its usefulness, functionality and pedagogical characteristics. Questions were divided into aspects of the software, some of them like interface, pedagogical characteristics, non-functional requirements, content, documentation. The checklists were confronted with a generalized checklist used by PROINFO, and can be seen that not generalized checklists present a better form of assessment of SEs.

Keywords: Software engineering, software testing, educational software

INTRODUÇÃO

O constante desenvolvimento da tecnologia tem sido um alterador na vida das pessoas, pois, estas procuram acompanhar todos os desafios propostos por ela. Neste contexto, o presente fato não se diferencia em relação à educação, que traz como finalidade primordial à formação e capacitação de cidadãos competentes e atualizados para a sociedade.

Atualmente o cenário pedagógico vem passando por várias modificações, sendo uma delas a utilização da tecnologia, de modo a propiciar uma maior motivação para os seus alunos.

Deste modo, a utilização de software apenas faz-se cada vez mais necessário, desde que venha implementado como uma ferramenta, ou melhor, seja voltado como auxilio nas atividades didático-pedagógicas, possibilitando o planejamento e a realização de projetos interdisciplinares que seja condizente com a realidade escolar.

De tal modo, verifica-se que o software educativo disponibiliza diversas maneiras para ser trabalhado variando desde representações a um conhecimento organizado e estruturado.

Neste contexto, o presente trabalho tem como finalidade a análise do software educativo, apresentando a checklist.

1. BREVE DEFINIÇÃO DE SOFTWARES

Antes de se adentrar ao presente tema propriamente dito, necessário se definir o que seja o processo de software buscando apresentá-la de forma clara e concisa para uma perfeita compreensão de todo o trabalho.

Para Sommerville (1995) esse processo se expressa como um conjunto de atividades, bem como resultados que associados vêem a produzir um produto tido como software.

Neste contexto Pressman (1997) retrata um processo de software como uma espécie de framework para todos os serviços que se sejam necessárias para a elaboração e construção de software visto como sendo de alta qualidade.

Para Thiry (2001) o processo acaba por definir quem deverá fazer determinada coisa, bem como se atingir o objetivo. Nessa perspectiva para autor o objetivo central será construir ou mesmo melhorar um software que já exista.

Através destas definições pode-se dizer que o processo necessariamente reunirá inúmeras atividades, além de que o processo tem como principal intuito o desenvolvimento de um software, ou seja, um produto de software.

Para Rezende (2002) o processo de desenvolvimento de um software pode ser comparado a uma receita de bolo, pois no seu decorrer pode-se adicionar ou mesmo suprir artefatos, para se chegar a uma correta execução.

Desta forma em um processo de software não há um exemplo apropriado a ser seguido, devendo-se analisar sua aplicação, objetivo, ambiente, e inclusive se um projeto especifico será mais vantajoso ou não para o caso.

1.1 Principais Fases de Um Processo de Software

Passa-se a analisar as fases de um processo de software. Assim segundo Schwartz (1975) retratava que as principais fases de um processo de produção de software são: Especificação de Requisitos; Projeto de Sistema; Programação; Verificação e Integração; Evolução.

Passa-se agora a analisar as fases de produção de um processo de software acima citadas:

- Especificação de requisitos - o mesmo é visto como uma expressão da necessidade ou mesmo de um requisito operacional para uma funcionalidade que deverá ser executada;

- Projeto de Sistema – o presente é visto como uma tradução dos requisitos, que pode ser comparado a uma descrição de todos os componentes que se fazem indispensáveis para a codificação do sistema;

- Programação / Codificação – o presente é visto como a produção de um código que venha a controlar todo o sistema, bem como realizar a computação que se diz necessária;

- Verificação e Integração – o presente é visto como a averiguação da satisfação de todos os requisitos iniciais produzidos pelo produto.

A fase de evolução é retratada por Sommerville (1995), conforme se demonstra:

- Evolução / Manutenção – o presente é visto como uma alteração em relação ao software, principalmente para atender as necessidades novas do usuário.

De a acordo com Pressman (1997) as etapas da produção de softwares são bem mais simples, podendo estas serem divididas em: desenvolvimento e o desenvolvimento.

Para Lech (2000) o desenvolvimento de produção de um software pode ser observado conforme o usado na empresa Microsoft, pois a maioria dos softwares acaba por serem desenvolvidos possuem apenas três fases, quais sejam: Planejamento; Desenvolvimento; Estabilização[1].

Necessário ressaltar que em relação à seqüência bem como aos números das fases não existe uma seqüência a ser seguida, nem mesmo um número a ser obrigatório. Diante deste fato, Rezende (2002) retrata que mesmo levando-se em consideração o ISSO 9000-3, não há um padrão de forma universal que se utilize, ou mesmo que seja aceito, podendo assim uma divisão ser dotada de diversas fases diferentes.

1.2 Arquitetura de Software

Um modelo para a descrição das arquiteturas de software. A descrição de uma arquitetura de software é formada por três elementos básicos:

- Um elemento de processamento é uma estrutura de software que transforma suas entradas em saídas necessárias.

- Um elemento de dados consiste nas informações convenientes para o processamento, ou em informações a serem processadas por um elemento de processamento.

- Os elementos de conexão são o amálgama que une as diferentes partes de uma arquitetura.

Por exemplo, uma única URL[2] é uma linha de comando na Internet utilizada para especificar um objeto na rede, tal como um arquivo ou newsgroup. Uma URL fornece um exemplo conciso dos elementos arquitetônicos. Tente digitar, por exemplo, a URL para encontrar os novos servidores e as novas ferramentas para Internet.

Durante o desenvolvimento de um sistema de software, é útil organizar as arquiteturas em família e associá-la as aplicações típicas. O beneficio disso é diminuir o tempo necessário para selecionar arquiteturas relevantes que satisfaçam aos requisitos de uma especificação de requisitos de software.

---

[1] A estabilização acaba por ocorrer quando as mudanças são visíveis e o número de erros encontra-se bem reduzidos, de forma a serem aceitáveis.
[2] URL (Uniform Resource Locator)

Conforme Kiely (1998, p. 145) um padrão de organização estrutural em uma arquitetura define o que é percebido por estilo arquitetônico. Um estilo arquitetônico caracteriza-se pelos seguintes detalhes:

- Tipos de componentes: Elementos de processamento utilizados para transformar dados;

-Tipos de conectores: Caminhos de controle e de dados entre os componentes;

- Restrições: Restrições quanto ao processamento, dados e formas permitidas de se unirem os componentes eletronicamente.

Cada estilo arquitetônico tem a aparência de uma caixa de ferramenta contendo instrumento útil para a construção de diferentes tipos de módulos de software. Como exemplo de diversas famílias de arquiteturas de software, podemos citar fluxo de dados, máquinas virtuais, específica para o domínio e retorno, independente do processo e sistema repositórios.

1.3 Validação de Um Projeto de Software

Existem cinco etapas no processo de validação e verificação de um projeto de software. No decorrer do processo de projeto, há um fluxo paralelo entre os requisitos e o projeto. Os engenheiros de requisitos e os projetistas de software corroboram os artefatos do projeto e monitoram as mudanças com maior facilidade.

As tarefas de seleção são desenvolvimentos das arquiteturas de software e demais artefatos de projetos elaborados no decorrer do processo de projeto são executados com os requisitos sempre em mente. A suposição subjacente e crucial de requisitos específicos possui único identificador, tornando-se possível relacionar os componentes de projeto a um requisito específico.

A análise da rastreabilidade requer que cada componente seja relacionado. Periodicamente, deve ser conveniente relacionar um componente de projeto específico, restabelecendo o fluxo entre eles. A análise da rastreabilidade fica mais fácil quando se tem uma estrutura de armazenamento durante o processo de projeto.

> O problema inicial solucionado pelo rastrear DPS é a construção de alguma forma de tabela de rastreamento. Isso pode ser feito através da construção de uma Matriz

> da Rastreabilidade de Requisitos (MRR) ou através da criação de um hipertexto da Web para artefatos de projetos, vinculado aos requisitos através de conexões. (MILLS, 1977, p. 77)

O estilo mais simples é o sistema de entrada X. Sempre que um X se apresenta na linha, traz o significado de que um artefato do projeto está em conformidade com um requisito. Sempre que a linha aparecer em branco, diz se que o componente do projeto é irrelevante.

Ambas as formas de rastreamento de projeto apresentam vantagens. A medição da rastreabilidade de projeto média com um valor mínimo torna possível a identificação de atributos de projetos que possuam ligações fracas com os requisitos. O requisito de rastreamento de projeto ponderado permite que os gerentes dos sistemas de software manifestem sua preferência.

Cada peso representa algum grau de importância de requisitos com pesos maiores deverão receber mais atenção. Essa abordagem afetará as estimativas de custos e a definição de orçamento. Ou seja, os requisitos com peso de valor mais alto serão considerados requisitos mínimos para um projeto de sistemas de software. Isso trata-se basicamente de uma ligação entre o conjunto de requisitos e seus custos para chegar a um orçamento mínimo de projeto.

1.4 Qualidade de Software – Garantias e Padrões

As atividades de garantia de qualidade projetam uma estrutura eficaz para atingir a qualidade de software. Esse processo de QA[3] envolve definir ou selecionar os padrões que devem ser direcionados ao processo de desenvolvimento e elaboração de software ou ao produto de software.

Esses padrões podem ser integrados em procedimentos ou processos que são aplicados durante o desenvolvimento. Os processos podem ser apoiados pelo uso de ferramentas que integrem o conhecimento dos padrões de qualidade. Existem dois tipos de padrões que podem ser constituidos como parte do processo de garantia de qualidade.

Padrões de produto: São os produtos que se aplicam ao produto de software em desenvolvimento. Eles agregam padrões de documentos, como a estrutura do documento de requisitos a serem produzidos; padrões de

[3] QA (Quality Assurancc)

documentação com um cabeçalho-padrão de comentário para uma definição de classe de objeto, e padrões de codificação, que definem como uma linguagem de programação que deverá ser utilizada.

Padrões de processo: São os padrões que escolhem os processos a serem seguidos durante o desenvolvimento de software. Eles podem incluir definições de especificação, processos de projeto e validação, e uma descrição dos documentos que devem ser elaborados no curso desses processos.

Existe uma relação muito estrita entre os padrões de produto e de processo. Os padrões de produto aplicam-se ás saídas do processo de software. E em muitos casos, os padrões de processo incluem atividades específicas de processo que asseguram que os padrões de produto que sejam seguintes. Será discutida a importante relação entre qualidade de produto e processo. Há uma série de razões pelas quais os padrões de software são importantes:

Eles fornecem um encapsulamento das melhores práticas ou, pelo menos, das mais adequadas. Essa habilidade geralmente é adquirida depois de um expressivo número de tentativas e erros. Constituir esse conhecimento dentro de um padrão evita a repetição de erros cometidos anteriormente.

Eles disponibilizam uma infra-estrutura em torno do qual o processo de garantia de qualidade poderá ser implementado. Considerando que os padrões incluem e exigem as melhores práticas, o controle de qualidade simplesmente garante que os padrões foram seguidos de maneira adequada.

Eles ajudam em termos de continuidade, quando o trabalho realizado por uma pessoa é assumido e continuado por outra pessoa. Os padrões asseguram que todos os engenheiros dentro de uma organização adotam as mesmas práticas. Conseqüentemente, o esforço de aprendizado é reduzido, quando se inicia um novo trabalho.

> O desenvolvimento de padrões de projeto de engenharia de software é um processo difícil e demorado. Organizações nacionais e internacionais, como US DoD e ANSI. Esses são padrões gerais, que podem ser aplicados a uma série de projetos. (MAXWELL, 1996, p. 126)

As equipes de garantia de qualidade que forem desenvolver padrões, geralmente, deverão co-relacionar os padrões organizacionais nos padrões nacionais e internacionais. Com a utilização desses padrões, que deve definir aqueles apropriados para sua organização.

Os engenheiros de software, algumas vezes, consideram os padrões como burocráticos e irrelevantes para a atividade técnica de desenvolvimento de software. Isso é particularmente provável quando os padrões exigem tediosos preenchimentos de formulários e registro de trabalho.

Os padrões de processo podem causar dificuldades, se um processo pouco prático for imposto á equipe de desenvolvimento. Esses padrões são, muitas vezes, simples diretrizes que devem ser interpretadas de modo compreensivo pelos gerentes de projetos individuais.

Não haverá nenhuma razão para prescrever uma maneira específica de trabalhar, se ela não for apropriada para o projeto ou para a equipe de projeto. Cada gerente de projeto detém, portanto, autonomia para modificar ou mesmo remodelar os padrões de processo, de acordo com as circunstâncias individuais.

1.4 Ciclo de Vida

O ciclo de vida acaba por direcionar a maior parte de todas as fases, bem como de todas as atividades que poderão ser seguidas para a perfeita produção de um produto de software[4], mas um ciclo não estabelece de forma clara os artefatos.

Segundo Thiry (2000) o modelo será uma coleção relacionada aos artefatos, sendo que os mesmos acabam por demonstrar a visão do sistema, deste modo o artefato demonstra ser as informações produzidas por um determinado participante, que pode ser um modelo ou documento. Importante salientar que os artefatos poderão ainda ter suas versões sempre controladas.

Existem modelos tidos como teóricos e que são desenvolvidos para descrever como as fases devem seguir e interagir. Deste modo um modelo

[4] Software, logiciel ou programa de computador é uma seqüência de instruções a serem seguidas e/ou executadas, na manipulação, redirecionamento ou modificação de um dado/informação ou acontecimento.

acaba por evidenciar o direcionamento relacionado às fases, ou seja, ciclo de vida, e não a descrição de como a atividade deverá ser sempre executada.

De acordo com Chikofsk (p. 225, 1994) A taxonomia de teste de software é parte de um estudo. A classificação ou colocação de várias formas de teste de software em conjuntos de procedimento de teste relacionados produz tal taxonomia. O teste de caixa preta pode ser também denominado de teste funcional.

O ponto inicial do teste de caixa preta é uma especificação ou código. No caso de ser código, os dados de teste instigam o software a verificar se são fornecidas as funções desejadas. O conteúdo da caixa é oculto. O software estimulado deve produzir resposta específica.

O teste de caixa branca é também denominado teste estrutural. Essa forma de teste estimula o projeto detalhado do software, em vez das funções. No teste estrutural, as instruções, os caminhos e as ramificações são observados para fins de execução correta.

Uma taxonomia dessas formas de teste é apresentada. E preciso estar ciente das limitações inevitáveis do teste. Existem duas fontes fundamentais para que ocorrem limitações. A intratabilidade é para testar exaustivamente até mesmo um programa pequeno poderia levar anos. E também a incapacidade de decisão.

Outra taxonomia importante do teste de software são as seguintes:

- Orientado á especificação: Aqui os dados de teste são desenvolvidos com base em documentos e a compreensão correspondente deve definir o comportamento de um sistema. As fontes aqui utilizadas abrangem especificações escritas reais, além de projetos de altos e baixos níveis. O teste orientado á especificação possui uma visão funcional do software podendo em alguns momentos ser denominado teste de caixa preta.

- Orientado á implementação: Nesta modalidade dos métodos de teste, os dados de teste são norteados pelas informações procedentes da implementação. Cada execução de um programa estuda um caminho específico em um programa.

- Orientado ao erro: O surgimento desta categoria é determinado pela presença potencial de erros no processo de programação.

Portanto as questões não técnicas deve ser concluído pelo programador anteriormente envolvido no processo de codificação. A pessoa

deve considerar alguma alternativa, como o teste independente, a troca de códigos entre os membros da mesma equipe de programação ou simplesmente deixar o teste a cargo de um usuário final.

Assim, Parnas (1972, p. 78) retrata que "diversas técnicas de teste podem ser classificadas de acordo com a visão aplicada nos sistemas, essas visões de software são classificadas com base no aumento do conteúdo de informações".

Cada uma dessas alternativas tem seus próprios prós e contras. Há também um questionamento de grande importância sobre a duração das atividades de teste, em que dois pontos de vista conflitantes e relevantes devem ser considerados, O primeiro é promulgado pelos projetistas do software que se esforçam para obter um produto sem faltas.

O segundo ponto é expresso pelos gerentes de projeto que podem e devem estudar a conveniência e eficácia do empreendimento. Antecedendo uma discussão detalhada referente diversas abordagens de teste, pode-se destacar as etapas principais sempre presentes em qualquer esquema de teste, apesar do caractere subjacente.

2. SOFTWARES EDUCATIVOS

O software educativo é considerado como uma ferramenta direcionada para dar auxilio o professor no desenvolvimento de todo o processo de ensino e aprendizagem, trazendo novos métodos e conhecimentos de maneira que venha a favorecer e fortalecer a aquisição do conhecimento por parte do aluno.

Verifica-se que esse favorecimento é o ponto fundamental para distinguir o software educativo dos demais programas existentes no mercado, principalmente os que possuem outras finalidades, como, por exemplo, empresarial, gerencial, etc.

Salienta-se que a utilização da presente ferramenta de modo eficiente é conveniente à realização de uma seleção de fatores e também conteúdos que o professor deseja e possa trabalhar com os seus alunos.

Passa-se a analisar os principais tipos de softwares educativos: tutorial, exercício e práticas, jogos, aplicativos. Importante salientar que a presente lista ainda possui outras espécies de softwares educativos, sendo, apenas apresentados os principais.

2.1 O Tutorial em Um Software Educativo

Segundo Oliveira, Costa e Moreira (2001), os tutoriais fornecem informações que vem a serem transmitidas em um diálogo entre o usuário, ou seja, aluno e o computador, tendo como a principal característica à apresentação de informações, como, por exemplo, resposta a uma pergunta, ou ainda a resolução de um determinado problema.

Desta forma, todas as informações apresentadas podem e devem tanto estimular como também motivar o aluno, ocorrendo assim um comprometimento em maior parte das ações relacionadas com a informação.

Deste modo, os programas tutoriais devem sempre introduzir novos conceitos, bem como, apresentar habilidades e atividades voltadas para a verificação da aquisição de todo o conteúdo. Assim, deverá ser utilizada como apoio ou mesmo reforço durante as aulas, orientando na revisão ou preparação de atividades, entre outros aspectos.

Segundo Valente (1999), o principal benefício dos tutoriais é o fato do computador trazer o material juntamente com outros tipos de características, que no papel não são possíveis de serem analisadas, como, por exemplo: som, animação, controle do desempenho do aluno. Nota-se que através destes benefícios, os programas tutoriais são utilizados com mais freqüência e viabilidade, pois, possibilita a introdução do computador na escola sem a ocorrência de grandes mudanças.

Os presentes softwares possuem como característica recursos motivacionais, bem como, controle da sequenciação do programa pelo próprio usuário, entre outros.

2.2 A Prática e Exercícios no Software Educativo

Os programas voltados para a prática e o exercício são considerados como a forma mais tradicional em que os computadores são utilizados atualmente na educação. Esse método, busca à aquisição de habilidades ou ainda práticas que ao se trabalhar com conteúdo que já seja conhecido pelo aluno, mas que o mesmo não tenha total domínio sobre ele. Todo este trabalho visa enriquecer o ensino em sala de aula, aumentando ou mesmo construindo as habilidades e criatividades básicas do aluno.

Geralmente são programas que possui como finalidade reforçar tanto os fatos como os conhecimentos que devem ser analisados em uma aula

expositiva ou mesmo em laboratório, por meio de perguntas e respostas (OLIVEIRA; COSTA; MOREIRA, 2001).

É possível citar como uma das principais e relevantes características da prática e exercícios o uso de feedback positivo que possibilita aprimorar o desempenho do aluno, através de repetição de exercício visando atingir os objetivos determinados no programa, entre outras.

Verifica-se que a vantagem do presente tipo de programa é que o professor poderá dispor de uma gama enorme de exercícios que poderá o aluno resolver sempre de acordo com o seu grau de conhecimento, bem como, interesse.

Assim, se o software educativo, além de apresentar o exercício também recolher as respostas averiguando qual fora o desempenho do aluno, então o professor terá realmente a sua disposição um dado muito importante sobre como o material é visto e classificado, e, como o mesmo está sendo absorvido pelos alunos.

Salienta-se que para alguns professores, estes dados não podem ser considerados inteiramente suficientes, pois não é fácil para o software educativo encontrar o motivo do erro ou mesmo do acerto do aluno.

Portanto, como o assunto está realmente sendo simulado exige-se muito conhecimento por parte do professor, pois, torna-se necessário analisar muitas outras variantes, não só o número de acertos e erros dos aprendizes.

Desta maneira, a idéia de que os programas de práticas e exercícios servem para suavizar a tarefa dos professores em relação a correção dos testes ou mesmo as avaliações não pode ser considerada real, pois, ocorre apenas à eliminação da parte mecânica da avaliação.

Constata-se que ter uma visão clara do que verdadeiramente está acontecendo com o processo de assimilação dos assuntos analisados em classe acaba por requerer uma visão bem mais aprofundada do desempenho dos alunos (VALENTE, 1999).

2.3 Jogos

Os jogos possuem como característica a existência de alguns recursos motivacionais, sendo estes voltados para despertar, bem como, manter e fixar a atenção do aprendiz.

Verifica-se que este tipo de software educativo acaba por ser uma fonte de recreação, direcionando para um determinado método de aprendizado. Ocorre, portanto, o envolvimento de elementos de desafios ou competições.

Os softwares de jogos possuem uma grande capacidade de capturar a atenção do aluno enquanto desenvolve sua atividade, principalmente por causa do seu aspecto colorido, além de ser, dinâmico e divertido.

Importante ressaltar que o presente software disponibiliza aos usuários uma enorme interatividade, de maneira a ampliar as relações sociais no ambiente de ensino, de modo, a prender o interesse dos alunos em relação a temas ditos como difíceis de serem apresentados de outra forma (OLIVEIRA, COSTA, MOREIRA, 2001).

Para Valente (1999) o principal problema com os jogos, é que em algumas vezes ele poderá afastar a atenção da criança em relação ao conceito envolvido no jogo direcionado para o ensino e aprendizagem, principalmente por se tratar de uma competição.

Oliveira, Costa e Moreira (2001) retratam que os jogos possuem como finalidade não apenas o entretenimento para o aluno, mas, sim, a influência e entusiasmo do desenvolvimento sócio-afetivo, bem como, cognitivo do indivíduo, pois, traz como a principal essência à aprendizagem com prazer e a criatividade com diversão.

2.4 Aplicativos

Os aplicativos são programas direcionados para algumas aplicações específicas, temos como exemplo, processadores de texto, gerenciadores de banco de dados, planilhas eletrônicas, etc.

Mesmo os aplicativos não tenham sido desenvolvidos para a utilização na área educacional, os mesmos permitem que os interessados os usem em diferentes ramos do conhecimento.

Valente (1999) retrata que os processadores de textos, bem como, as ações do aluno devem sempre ser analisado seguindo o determinado ciclo: descrição, execução, reflexão, depuração, descrição.

Portanto, quando o aluno está digitando um texto no processador de texto, a relação existente entre ele e o computador é considerada mediada pelo idioma, bem como, pelos comandos de formatação.

Importante ressaltar que apesar dos processadores de texto expressarem o pensamento, ele não poderá executar o conteúdo do mesmo, e fornecer um feedback com o resultado, bem como, do seu significado para o aluno.

Assim, verifica-se que a única possibilidade, em relação à reflexão, é a comparação das idéias originais do formato com o resultado apresentado, não possibilitando nenhuma margem para a reflexão e também depuração de todo o conteúdo.

Deste modo, o processador de textos não possui características que venham a auxiliar o processo de construção do conhecimento, bem como, da compreensão da idéias.

2.5 A Checklist na Qualidade de Softwares Educativos

Necessário deixar claro que não é fácil a realização da avaliação de software educativo, pois, é indispensável dispor de diversos critérios que permitam a realização de uma análise totalmente eficaz em relação ao valor e viabilidade educacional e também computacional de todos os materiais.

Segundo Gladcheff, et al (2001) a característica de funcionalidade acaba por evidenciar o conjunto de ações voltadas para atender a todas as exigências explícitas ou implícitas para a finalidade que se destina o produto final. Sendo assim, a confiabilidade é na realidade a capacidade do produto em manter seu verdadeiro desempenho no decorrer do tempo, principalmente em condições estabelecidas.

Para Anjos e Moura (2004) no caso do software educativo a usabilidade é um meio facilitador para se usar o produto. Deste modo, verifica-se a existência de variadas técnicas e métodos para a avaliação de um software educativo.

De tal modo, a metodologia mais utilizada geralmente é o conjunto de questões específicas, ou mesmo preestabelecidas, que possuem como objetivo a condução do processo de avaliação, ou seja, *checklist*.

Necessário que o software educativo apresente algumas características para que seja aplicado e usado, redundando em um melhor desempenho no decorrer do processo de aprendizagem.

Portanto, o conteúdo deverá sempre que possível apresentar uma forma objetiva, dando prioridade a interatividade e a criatividade, proporcionando um feedback. Mas, por outro lado, ele sempre deve ser um

estimulante, pois, deverá ser provocativo e desafiador, de modo, a prender a atenção do aprendiz (MORAES, 1997).

Segundo Campos (2001) ao se analisar a qualidade de um software voltado para a educação, é eficaz sempre considerar atributos que evidenciem a facilidade e simplicidade de utilização do mesmo, bem como, a existência de meios e recursos que podem facilitar a interação do usuário, assim como, a capacidade do software em se adaptar a necessidades e preferências do usuário ao ambiente.

Assim, as principais características do software no *checklist* são:

a) funcionalidade – se divide em: 1) clareza dos conteúdos: as informações de todos os conteúdos devem ser claras e principalmente consistentes, facilitando, então a sua compreensão.; 2) assimilação e acomodações:o tratamento em relação à apresentação deve sempre reverenciar a construção do conhecimento; 3) carga educacional: indispensável a verificação em relação à quantidade de todas as informações apresentadas pelo software. Deste modo, tanto a falta como o excesso de informação torna-se totalmente prejudicial à aprendizagem; 4) avaliação do aprendizado: são apresentados recursos para a verificação do aprendizado (MORAES, 1997); 5) Motivação; necessário a utilização de recursos motivacionais que venham a despertar a atenção do aprendiz no decorrer da utilização do software; 6) Adequabilidade: a necessidade da adequação do vocabulário (CAMPOS, 1995); 7) Uso de cores, imagens e sons: necessário a utilização de cores, ilustrações, animações, e também recursos sonoros para despertar o interesse do aprendiz (SOUZA, 2004); 8) Randômico: necessário a existência de geração randômica de atividades (CAMPOS, 1995); 9) Robustez: importante para manter o processo correto em relação às ações inesperadas; 10) Afetividade: acaba por proporcionar uma relação totalmente agradável para o aprendiz ao longo de seu processo de aprendizado (CAMPOS, 2001), etc.

b) Confiabilidade – 1) Maturidade: necessário analisar a freqüência de falhas; 2) Tolerância à falhas: necessário observar quando ocorre falhas a reação; 3) Recuperabilidade: como é capaz de recuperar os dados quando ocorre à falha (ANJOS E MOURA, 2004)

c) Usabilidade – 1) Facilidade de leitura: necessário observar se a leitura em tela é fácil; 2)Clareza: os comandos desenvolvidos para o usuário são claros; 3) Feedback: ocorre o fornecimento de feedback (CAMPOS, 1995)

d) Eficiência – 1) Armazenamento – necessário analisar o armazenamento das respostas; 2) Adaptável: verifica-se a adaptabilidade ao nível do usuário; 3) Correção: ausência de qualquer tipo de erros no processamento do programa (CAMPOS, 1995); 4) Tempo: avalia-se o tempo de resposta, assim como a velocidade de execução (ANJOS E MOURA, 2004)

Desta forma, a avaliação negativa de determinadas características de uma checklist não pode significar que o software educativo venha a ser considerado como de má qualidade, pois, demonstra um peso bem menor em todo esse processo geral de avaliação. Assim, verifica-se que para cada software os critérios a serem levados em consideração são totalmente diferentes, devendo ser definidos, portanto, pelo avaliador. Com isto, o resultado da avaliação vem ser fornecida de forma quantitativa, especialmente por meio do resultado obtido por meio da verificação (DAVIDONE, 1987).

2.6 Checklists Específicas

Por meio do levantamento das principais características que os softwares educativos devem possuir, pode-se, então, migrar para o estudo das especificidades de cada modalidade de software educativo, baseadas nas propostas pedagógicas e definições de cada um, como descrito anteriormente.

Mesmo este trabalho ainda estar em período de finalização, pôde-se elencar alguns aspectos específicos relevantes quanto às modalidades de software educativo contidas neste trabalho, algumas como: linearidade/não-linearidade, interface, proposta pedagógica, requisitos não funcionais.

Claramente que em todos os softwares educativos necessita-se de se aplicar a checklist que checa as questões apontadas no capítulo 2.5, juntamente com a checklist que contém as questões específicas para cada modalidade de software educativo.

RESULTADOS INICIAIS

Por meio do levantamento de alguns aspectos específicos da modalidade de tutorial de software educacional, pode-se perceber que a análise de um tutorial tornou-se mais concreta e objetiva. A checklist das características principais do software, por meio do capítulo 2.5, sanaram e resolveram problemas de possíveis aplicações de certos tutoriais, assim como questões de ordem específica da modalidade puderam ajudar no melhor formato de utilização daquele software educativo.

Estes resultados tornaram-se mais visíveis quando comparados a resultados obtidos de cheklists generalizadas produzidas pelo PROINFO.

CONSIDERAÇÕES FINAIS

A globalização veio a trazer o desenvolvimento tecnológico, proporcionando, assim, enormes transformações na sociedade e também na vida das pessoas.

Tal fato afetou também a escola, sendo intensificado o desenvolvimento de softwares educativos que veio a incentivar a utilização das tecnologias da informação, até mesmo, da comunicação em meios escolares, facilitando e diversificando as metodologias antes utilizadas, beneficiando o aprendizado de um maior número de alunos.

O software educativo deve ser utilizado pelo professor como uma ferramenta de apoio a todas as atividades pedagógicas, não podendo de forma isolada ser aplicado no processo de alfabetização.

Por meio da estratégia pedagógica, o software educativo quando aplicado acaba por motivar o aluno para a aprendizagem. Assim, ao ser utilizado pelo professor com outras informações, deve exercitar a capacidade de pensar, criticar e também refletir sobre todas as suas ações.

Importante observar que para que um software desenvolva o seu verdadeiro potencial didático, torna-se conveniente e quase que obrigatório à mediação do educador. Assim, é através do seu trabalho e de sua forma de trabalhar que as atividades serão motivadoras para os seus alunos.

Assim, nota-se que o software educativo é uma ferramenta de grande valor tanto para os professores quanto para seus alunos, principalmente no que se refere à motivação e participação mutua tanto dos alunos como do professor.

REFERÊNCIAS BIBLIOGRÁFICAS

ANJOS, L. A. M.; MOURA, H.P. **Um modelo para avaliação de produtos de software.** Revisado em 2004. Disponivel em: HTTP://php.cin.ufpe.br/~laps/laps/biblioteca.php. Acesso em: 13 abr. 2010

CAMPOS, G.H.B. **A qualidade em software educacional.** Revisado em 1995. Disponivel em: HTTP://www.casadaciencia.ufrj.br. Acesso em: 13 abr. 2010

_____. **Como avaliar um software educacional.** Revisado em 2001. Disponivel em: HTTP://www.casadaciencia.ufrj.br. Acesso em: 13 abr. 2010

CHIKOFSKY, E. CROSS, j.H. **II Reverse engineering**. In Encyclopedia of software Enginering. J.J Marciniak, Ed. Wiley, Nova York. 1994.

GLADCHEFF, A.P; ZUFFI, E; SILVA, M. **Um instrumento para avaliação da qualidade de softwares educacionais de matemática para o ensino fundamental.** In: VII Workshop Sobre informática na escola. Fortaleza (s.d), 2001

KIELY, D. **Are components the future of software?** IEEE Computer. 1998.

LEACH, Ronald J. **Introduction to Software Engineering**. Florida: CRC Press LLC, 2000.

MAXWELL, K. D., et al. **Software developmente productivity of European space, military and industrial applications**. 1996.

MAXWELL, K. D., et al. **Software developmente productivity of European space, military and industrial applications**. 1996.

MILLS, H.D. **Top-down programming in large systems**. In. Debugging Techimiques in Large Systems. 1977.

MORAES, M.C. **Informática educativa no Brasil: uma história vivida, algumas lições aprendidas.** Revista brasileira de informática na educação, Florianópolis, n.1, set. 1997

OLIVEIRA, C.C.; COSTA, J.W; MOREIRA, M. **Ambientes informatizados de aprendizagem: produção e avaliação de software educativo.** Campinas: Papirus, 2001

PARNAS, D. L, **A technique for software module specification with examples**. Comunications of the ACM. 1972.

PRESSMAN, Roger. S. **Software Engineering**: A practitioner's approach. New York: Ed. McGraw-Hill, 1997.

REZENDE, Denis A. **Engenharia de Software e Sistemas de Informação**. Rio de Janeiro: Brasport, 2002.

SOMMERVILLE, Ian. **Software Engineering**. Lancaster University, Lancaster: Ed.Addison-Wesley, 1995.

SOUZA, P.C.; BOMFIM, S.S; SCHENEIDER H.N. **Avaliação ergonomica de softwares educativos infantis: uma abordagem baseada no uso de cheklist e tercnicas empíricas.** Revista do Mestrado em Educação, São Cristóvão, v.9, jul/dez. 2004

THIRY, Marcello. **Processo de Desenvolvimento de Software com UML**, jun,2001. Disponível em: <http://www.eps.ufsc.br/disc/procuml/>. Acesso em: Acesso em 19 mai. 2008.

VALENTE, J.A. O computador na sociedade do conhecimento. São Paulo: NIED, 1999