Engenharia de Software no Desenvolvimento de Software Educacional Hipermídia

Glauco Cabral Monteiro de Castro (1)
Departamento de Sistemas
Diretoria de Abastecimento
Marinha do Brasil
glauco@netgate.com.br
glauco@dabm.mar.mil.br

Teresa Cristina de Aguiar (2)
Departamento de Ciência da Computação
Universidade Federal Fluminense
cris@dcc.uff.br

Resumo : A rapidez com que as aplicações tem incorporado recursos das mais variadas mídias de comunicação e ampliado suas áreas de utilização, tem provocado um lapso na esfera da Engenharia de Software em elaborar procedimentos que acompanhem com a mesma velocidade e aplicabilidade a utilização destes recursos.

A utilização da informática como ferramenta cognitiva na área de Educação, é um assunto amplo e contraditório e tem provocado questionamentos importantes, que vão desde a forma com a qual o conteúdo é apresentado ao aluno, chegando até os objetivos educacionais a serem atingidos com a utilização do software.

Com base nestas motivações, este trabalho tem como objetivo principal propor um processo de desenvolvimento para apoiar a tarefa de elaboração de sistemas educacionais em multimídia. São detalhadas as fases desse processo apresentando-se o seu propósito, métodos a serem utilizados e produtos gerados.

Palavras-Chave : Informática Educativa; Engenharia de Software

Abstract : Computer applications have incorporated varying communication media resources and a enlarged their areas of use in such a rapid pace that Software Engineering has not been able to elaborate procedures that accompany the use of these resources with the same velocity and applicability.

The use of information technology as a cognitive tool in education is a vast and contradictious matter that has provoked important questions ranging from the way of presenting contents to student to the establishment of educational objectives to be reached with the software use.

Based on such motivations, this work has a main objective of proposing a development process to support the task of multimedia educational systems elaboration. The process phases are detailed showing its goal, required methods and delivered products.

Keywords : Educational Informatic; Software Engineering

(1) Mestrando em Ciência da Computação pela Universidade Federal Fluminense
(2) Professora Orientadora

1.0 Introdução

Este trabalho tem como objetivo principal propor um processo de desenvolvimento para apoiar a tarefa de elaboração de sistemas educacionais em multimídia. São detalhadas as fases desse processo apresentando-se o seu propósito, métodos a serem utilizados e produtos gerados.

2.0 O Processo : Etapas, Métodos e Produtos Gerados

O processo é composto das seguintes atividades :

- Preparação : Formação da Equipe de Trabalho, Escolha dos Modelos de Desenvolvimento e o Projeto Instrucional;
- Prototipação : Análise, Projeto, Implementação, Teste e Avaliação; e
- Implantação : Manutenção e *Up-Grade*, Controle de Versões, Suporte Técnico, Rotina de Configuração e Distribuição.

2.1 O Ciclo de Vida

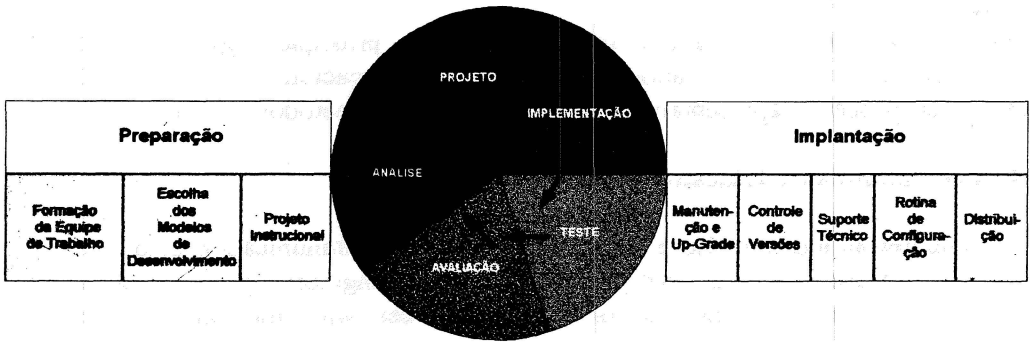

Figura 1: Ciclo de Vida proposto

O ciclo de desenvolvimento de aplicações educacionais possui, pela grande interatividade do *software* de autoria, um perfil de prototipação evolutiva.

O desenvolvimento se inicia com atividades preliminares com características de preparo de um ambiente propício a este desenvolvimento, como : a formação da equipe de trabalho e a escolha dos modelos a serem utilizados durante o processo de construção do *software*.

A prototipação evolutiva engloba atividades típicas de procura de requisitos e a implementação destes, a cada versão do protótipo novas funcionalidades são acrescentadas assim como novos temas são incorporados à aplicação.

As atividades de implantação também são dispostas fora das iterações de prototipação, caracterizando uma prontificação para o uso.

Nas seções que se seguem, cada uma das atividades pertencentes às iterações serão definidas com detalhes.

2.2 As Etapas de Desenvolvimento

Para descrevermos e melhor visualizarmos as etapas de desenvolvimento, vamos utilizar o modelo *Use Case*, por entendermos ser este modelo de simples construção e entendimento. Apesar do diagrama *Use Case* também fazer parte do grupo de modelos utilizados pelo método proposto, nesta seção ele se presta apenas como uma ferramenta gráfica de entendimento e é claro, não possui nenhuma função adicional de desenvolvimento da aplicação.

2.3 Diagrama de Casos de Uso

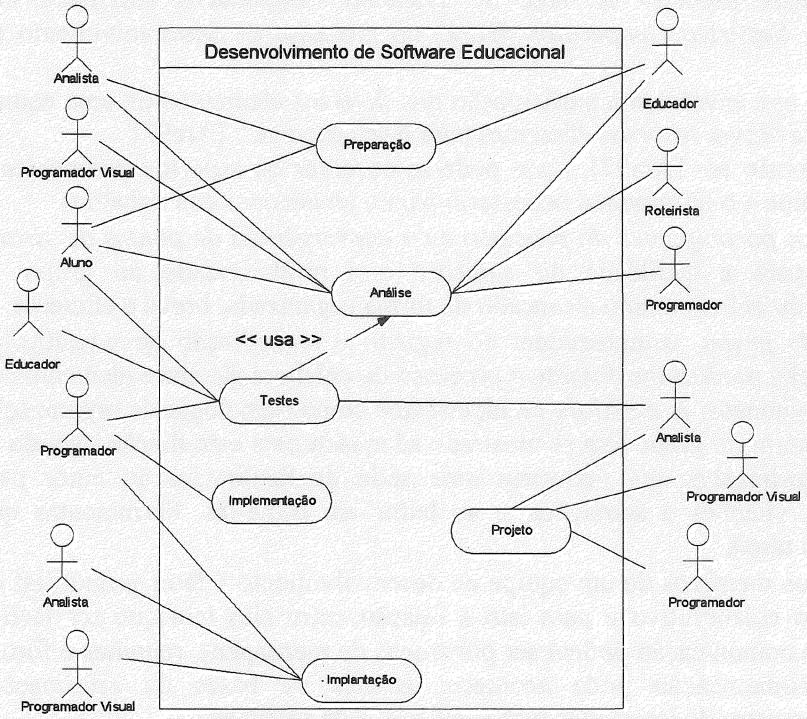

Figura 2: Caso de Uso : Desenvolvimento de *Software* Educacional

2.4 Descrição dos Casos de Uso

1) Caso de Uso : Preparação
Atores : Analista, Aluno, e Educador
Descrição :

1.1 - A Formação da Equipe de Trabalho - A tarefa de desenvolvimento de *software* com estas particularidades, envolvem tantos fatores, que não pode ficar dedicada a uma equipe com apenas um tipo de *skill* profissional, seja ela formada por pessoas do domínio do conhecimento que será modelado, ou apenas por educadores e alunos, ou então simplesmente por profissionais da área de Engenharia de *Software*, tais como analistas de sistemas e programadores. Em [Lin95] verificamos que podemos acrescentar a estes grupos de profissionais, aqueles ligados a área de multimídia, tais como :

- Artistas gráficos, que podem auxiliar a projetar o visual e aparência da apresentação. Embora não seja necessário ter experiência em programação multimídia, os artistas gráficos devem ser bem versados nas particularidades de projetar para multimídia.
- Programadores com experiência sólida em produção de multimídia.
- Escritores que não ficam intimidados pelos desafios de roteiros não-lineares e interativo.

A proposta para que estas diferenças sejam minimizadas é compor uma equipe de autoria do *software* multidisciplinar. A autoria consiste em fornecer mecanismos que possibilitem o planejamento da atividade do grupo de trabalho e a representação da mesma em um ambiente compartilhado[Ben97]

Equipes com esta heterogeneidade tem se tornado tão freqüentes, que esforços tem sido envidados no sentido de pesquisar aspectos da área de Trabalho Cooperativo Auxiliado Por Computador (CSCW – *Computer Suported Cooperative Work*) no contexto de desenvolvimento de *software*.

"Compreender as questões que envolvem a participação dos diversos elementos de uma equipe de *software* é o primeiro passo para prover recursos eficientes para o seu suporte". [Ara97]

Sugerimos o esquema proposto em [Ara97], onde pode-se observar os aspectos pertinentes à interação entre os membros da equipe e o ferramental necessário para o andamento dos trabalhos.

- *O entendimento entre os participantes do processo e/ou convergência de pontos de vista* é uma questão crucial para a qualidade do *software* e a produtividade do grupo. A produtividade depende do entendimento alcançado de forma organizada, breve e eficiente.
- Suportes à *memória do grupo*, compreendem no registro e recuperação de artefatos de *software* e documentação produzidos durante o processo de colaboração entre elementos de um equipe de desenvolvimento. A metáfora de hipertextos como tecnologia de organização e disseminação de memória do grupo tem se mostrado adequada para esta função, devido às suas características não-lineares que permitem uma série de facilidades ao autor para especificar referências, citações e anotações, e ao leitor em trilhá-las. Ferramentas que fornecem suporte a esta tarefa :
- A *Comunicação* entre os membros de um equipe de desenvolvimento é fator primordial no andamento do processo colaborativo e para isto a ligação entre eles tem que ser fácil e produtiva. Os canais de comunicação podem ser por trocas de mensagens, reuniões e fóruns de discussão. Uma comunicação pode acontecer através de bases de informações compartilhadas com o registro de interações realizadas por cada participante.
- A *Coordenação* refere-se ao acompanhamento e gerenciamento constante das atividades realizadas pelo grupo. A *comunicação* e a *percepção* são importantes para que aspectos da coordenação sejam discutidos e resolvidos em tempo hábil.
- A *Percepção* é definida [Ben97] como a capacidade de entender a finalidade e a influência da participação individual na realização da tarefa do grupo.

1.2 - Escolha dos Modelos de Desenvolvimento - Esta atividade caracteriza-se pela escolha dentre os diversos diagramas e modelos de desenvolvimento, utilizados no processo de construção de aplicações de uma forma mais genérica.

1.3 - Projeto Instrucional - Apenas os aspectos de interação e motivação são insuficientes linhas de ação para os educadores na análise do *software* multimídia para educação. O Projeto Instrucional deve possuir elementos construtivistas no sentido de buscar obter da relação do sujeito com o objeto manipulado, uma rica fonte da formação do conhecimento. É o conceito de *brinquedo* onde Vygotsky propõe um paralelo entre o brinquedo e a instrução escolar, no qual ambos reduzem a distância entre o conhecimento atual do sujeito e o seu conhecimento potencial. Esta atividade será desenvolvida pelos educadores e alunos que compõem a equipe de desenvolvimento.

Os seguintes eventos do aprendizado [Gag85] podem servir de um *framework* para um bem sucedido projeto multimídia : **(1)** obter a atenção do aluno, **(2)** informar os objetivos de aprendizado de cada lição, **(3)** recordação de assuntos prioritários, **(4)** apresentação de material estimulante, **(5)** prover um guia de aprendizado, **(6)** prover *feedback*, **(7)** melhorar a retenção e a transferência de aprendizado e **(8)** descobrir níveis de performance de aprendizado e aumentar estes níveis.

Devemos nesta atividade, analisar como esses eventos devem ser alcançados.

2) Caso de Uso : Análise
Atores : Analista, Aluno, Roteirista, Educador, Programador e Programador Visual.
Descrição :

A etapa de Análise, conforme visto em [Rum94]:

"*ocupa-se com o delineamento de um preciso, conciso e correto modelo do mundo real*".

O propósito da Análise é modelar o sistema do mundo real de forma que possa ser entendido. Para isso é preciso levantar os requisitos, examina-los, analisar sua implicações e redefini-los rigorosamente.

Esta fase, comum a todos os métodos de desenvolvimento de *software*, representa o estudo da área de conhecimento objeto da aplicação a ser desenvolvida e a conseqüente captura dos requisitos necessários ao *software* para que este atenda aos anseios dos seus utilizadores.

Por requisitos do sistemas, entenda-se as funcionalidades que deverão ser representadas no *software* multimídia, a fim de garantir o entendimento e a identificação com o mundo real, por parte do público usuário da aplicação

O processo de definição dos requisitos segundo J.W. Brackett (citado em [Fio98]), compreende as seguintes atividades :

- Identificação dos Requisitos – Elicitação dos requisitos através da interação com as pessoas envolvidas;
- Identificação das Restrições de Desenvolvimento de *Software* – Descoberta de restrições no processo de desenvolvimento, como por exemplo o custo do *hardware* envolvido;
- Análise de Requisitos – Avaliação dos problemas potenciais;
- Representação de Requisitos – Através de textos, diagramas, modelos ou regras de prototipagem;
- Comunicação de Requisitos – Apresentação aos envolvidos no desenvolvimento dos resultados da definição dos requisitos;

- Preparação para Validação de Requisitos de *Software* – Estabelecimento de critérios de aceitação e técnicas para garantir que o *software*, quando produzido, satisfaça os requisitos do cliente e do desenvolvedor;
- Gerenciar o Processo de Definição de Requisitos – Acompanha todo o ciclo de vida do *software*.

2.1 –Lista de Atores Candidatos – A lista de atores candidatos, é um mecanismo inicial que irá auxiliar na elicitação dos requisitos. A partir dos atores, de suas funções no processo, de que eles precisam para efetuar os seus trabalhos é que iremos fazer o levantamento de requisitos. A seguir propomos uma lista dos atores que participarão das fases do método proposto.

Lista de Atores Candidatos	
Atores	**Descrição**
Analistas	Desenvolvedores responsáveis na modelagem lógica do *software*
Educadores	Professores da disciplina de Engenharia de *Software* e Multimídia, que irão utilizar o *software* como suporte às suas aulas.
Alunos	Serão os utilizadores do *software* educativo, mais especificamente, todos os demais atores, porém em uma atividade de aprendizado
Programadores	Desenvolvedores especializados no *software* de autoria adotado.
Roteiristas	Profissionais responsáveis em seqüenciar os eventos levantados tornando mais fácil a navegação entre eles, contribuindo para o aprendizado
Programadores Visuais	Especialistas na manipulação dos recursos multimídia.

2.2 - Identificação dos Requisitos – Esta atividade será desempenhada por todos os membros da equipe, ainda que os especialistas sobre o tema a ser desenvolvido sejam os educadores e os alunos. A importância da participação dos demais membros da equipe nesta fase do desenvolvimento, se explica pela necessidade de nivelamento e padronização do entendimento de todos sobre o tema a ser abordado.

O surgimentos dos métodos apoiados no paradigma orientado em objetos, no início dos anos 90, consolidou a idéia de que a captura dos requisitos do sistema deve ser através da ótica do usuário. Dentre o conjunto de métodos que surgiram desde então, uma técnica de modelagem de requerimentos mostrou-se mais eficiente e interativa com o usuário, esta abordagem é chamada de "OBJECTORY" *(Object Factory*. Engloba *Use Cases* na análise dos objetivos do sistema e aplicação de engenharia de reversa. *Use Cases* representam a visão externa do sistema com relação a seus usuários e mostram o sistema e suas interações com entidades externas (atores) destacando suas ligações, convencionalmente chamadas de fluxos e mostrando o corpo do diagrama (algo como o diagrama de contexto) que são denominados "*Use Cases*". Uma característica importante é que nós podemos discutir com os usuários ou com a equipe de desenvolvimento com facilidade e encontrar novos requerimentos para o sistema.

A identificação dos requisitos será auxiliada pela utilização de recursos gráficos (Diagrama de Casos de Uso) e textuais (Descrição de Cenários) para representação dos requisitos elicitados. Os assuntos a serem abordados serão representados por *use cases*.

2.3 – Identificação das Restrições do Desenvolvimento de *Software* – Esta atividade será desempenhada por todos os membros da equipe. Cada membro da equipe poderá opinar sobre restrições ou limitações no processo de desenvolvimento.

2.4 – Representação dos Requisitos – Esta etapa da Análise dos requisitos consiste na construção de modelos que representem, cada qual, aspectos diferentes dos requisitos identificados no passo 2.2. Estes modelos são : o Modelo de Objetos ou Estático e Modelo Dinâmico.

2.4.1 – Modelo de Objetos - O primeiro modelo a ser construído é o Modelo de Objetos. Este modelo mostra a estrutura estática dos dados e a organiza em parte manipuláveis, ele será a base para construção de uma base de dados da aplicação. A aplicação educativa poderá ou não se utilizar de uma base de dados. Os dados que pertencerão a esta base dependerão de algumas questões, tais como :

- Quais serão os tipos de objetos que serão utilizados na aplicação (textuais, imagens, sons, animações...) ?
- Haverá algum controle de acesso dos usuários da aplicação ?
- Serão fornecidos testes sobre os assuntos abordados aos usuários da aplicação ?
- Como esta disposta a estrutura da aplicação ? Em capítulos, tópicos e sub-tópicos, por exemplo.

Para representar a estrutura da Base de Dados da aplicação poderá ser utilizado o Diagrama de Classes de qualquer método orientado a objetos, pois apresenta facilidades para representação de objetos dos mais variados tipos além de possuir o suporte de várias ferramentas CASE.

2.4.2 – Modelo Dinâmico – A modelagem dinâmica consiste na representação da ação do tempo nos elementos estáticos estruturados no passo anterior. Este modelo, normalmente é considerado secundário em aplicações convencionais, com uma baixa interatividade com o usuário. Porém em aplicações hipermídia, onde a navegação não seqüencial é um dos diferenciais, este tipo de modelagem ganha em importância, quando utilizada para representar esta navegação.

2.5 – Validação dos Requisitos – A representação gráfica produzida nos passos anteriores servirão para uma validação junto ao usuário final no sentido de verificar se todos os requisitos estão listados e serão representados. Nesta reunião de validação é conveniente a participação apenas do analista, educadores e alunos.

3) Caso de Uso : Projeto
Atores : Analista, Programador Visual, Programador, Roteirista, Professor e Aluno
Descrição : As fases das iterações de prototipação tendem a se contrapor, porém com o intuito de melhor definir suas diferenças, as sub-atividades da fase projeto possuem o objetivo de complementar os modelos criados na fase anterior, acrescentando os requisitos não funcionais. As atividades normalmente desenvolvidas na etapa de projeto, referentes a arquitetura do sistema, não serão aplicáveis para o tipo de *software* objeto deste trabalho. Estas atividades se referem a *software* de médio e grande porte, que são dispostos em vários processadores e com uma complexidade no média ou superior.

3.1 – Escolha da Metáfora - Esta atividade será desempenhada pelos membros da equipe, com o perfil artístico, tais como roteiristas e programadores visuais. Uma metáfora é uma forma alternativa de se pensar a respeito de alguma coisa. Pode-se dizer que constituem um relacionamento parcial entre dois

conceitos. É uma abstração da realidade, isto é, não significa que ela tenha todas as características do mundo real. É semelhante mas não idêntico.

A escolha da metáfora que melhor representará o tema a ser abordado é uma atividade muito importante, pois deverá apresentar os assuntos de forma eficiente e atraente, facilitando assim o aprendizado. Estes são exemplos de metáforas freqüentemente utilizadas : Mesa de trabalho, Enciclopédias, Cenários, Sala de controle ou central de operações, Livros textos e Excursões ou Roteiros.

3.2 – Projeto de Interfaces – Esta atividade ganha em importância no tipo de *software* em questão, pois deve oferecer um bom nível de interação e um planejamento didático.

Alguns princípios foram escolhidos em [Dra98] para guiar o projeto de interfaces em um ambiente de hipertextos :

- *Identificar a metáfora* ;
- *Identificar todos os objetos que pertencem ao sistema ;*
- *Identificar todas as ações/funções que podem ser executadas sobre estes objetos. Classificar estas ações em ações genéricas, ações explícitas e funções de controle* ;
- *Identificar modificadores e filtros que possam selecionar estes*;
- *Identificar estratégias de escolha que podem permitir ao usuário acompanhar uma tarefa*;
- *Identificar classificações laterais;*
- *Identificar o formato dos objetos, partes de objetos, menus ;*
- *Identificar listas de objetos*;
- *Identificar ações reativas que possam ser executadas em uma lista ou conjunto de objetos*;
- *Identificar processos ou funções que compartilhem as informações ;*
- *Identificar todos os estados de interação com o usuário;*
- *Identificar helps on-line necessários através do sistema*;
- *Identificar todas as condições de erro*;
- *Identificar o layout da tela.*

Em [Bor97] encontramos uma proposição de uma janela de navegação, com vários elementos, cada qual com a sua função específica :

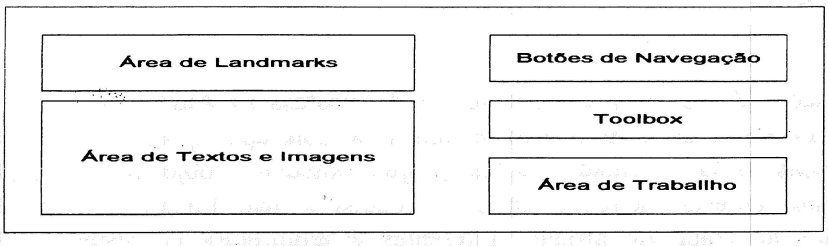

Figura 3: Proposta de uma janela de navegação [Bor97]

3.3 – Projeto de Objetos - Esta atividade se caracteriza por uma descrição mais refinada das classes enumeradas na fase de análise. Alguns dos aspectos observados nesta etapa :

- Neste momento as operações que surgem do projeto de interface e do modelo dinâmico são incorporadas à estrutura dos dados;
- Criação dos algoritmos para implementação das operações;

- Projeto dos dados, ou seja, o projeto físico da estrutura lógica dos dados visualizada na fase de análise.

3.4 – Modelagem dos Requisitos Não Funcionais (RNF) -

A descoberta de RNF deve ser feita em momentos diferentes aos destinados à elicitação de requisitos funcionais, de maneira manter o engenheiro de *software* concentrado no problema a ser resolvido.

Com relação aos RNF, em [Sta88] é proposto um *checklist* genérico:

Qualidade de *Software*		
Objetivos	**Fatores**	**Subfatores**
Confiabilidade da Representação	Legibilidade	Clareza
		Concisão
		Estilo
		Modularidade
	Manipulabilidade	Disponibilidade
		Estrutura
		Rastreabilidade
Utilizabilidade	Manutenibilidade	
	Operacionalidade	Oportunidade
		Amenidade ao Uso
	Portabilidade	
	Reusabilidade	
	Eficiência	
	Rentabilidade	
	Avaliabilidade	Verificabilidade
		Validabilidade
Confiabilidade Conceitual	Fidedignidade	Precisão
		Completeza
		Necessidade
	Integridade	Robustez
		Segurança

Figura 4: Quadro demonstrativo de requisitos não funcionais.

4) Caso de Uso : Implementação
Atores : Analista, Programador e Programador Visual
Descrição : Implementação deve ser uma parte mecânica e relativamente de menor importância do ciclo de desenvolvimento, porque todas as decisões difíceis devem ser tomadas durante o projeto. A linguagem influencia as decisões de projeto até um certo ponto, mas o projeto não deve depender de pequenos detalhes da linguagem de programação. A programação apesar de ser idealmente uma tarefa quase mecânica, pode ser embasada em diretrizes de estilo baseadas em objetos nas seguintes categorias [Rum94]: **(1)** Reusabilidade, **(2)** Extensibilidade, **(3)** Robustez e **(4)** Programação em Grande Escala .

O *software* educacional multimídia pode ser implementado utilizando-se diversos *software* de autoria multimídia.

5) Caso de Uso : Teste
Atores : Analista, Programador, Educador e Aluno.
Descrição : Para esta atividade, e sugerido o modelo de testes do método *Objectory (Object Oriented Software Engineering* - OOSE [Jac92]).

6) Caso de Uso : Avaliação
Atores : Analista, Educador e Aluno.
Descrição : Esta etapa nas diversas iterações da prototipação evolutiva proposta no ciclo de vida apresentado, marca o fim de uma versão do produto. A quantidade de versões a serem entregues deve ser acordada no início do desenvolvimento com o usuário, a fim de se tentar evitar um número excessivo de ajustes que irão impactar no prazo de entrega do produto. Basicamente esta etapa envolvem três atividades : **(1)** Apresentação da aplicação ao usuário, **(2)** A interação da entre usuário e a aplicação e **(3)** O registro das sugestões para a incorporação destas na próxima versão do produto.

7) Caso de Uso : Implantação
Atores : Analista, Programador e Programador Visual
Descrição :

7.1 - Manutenção e Up-Grade - Como todo *software*, o educacional, também precisa ser atualizado periodicamente com novas versões, talvez com uma freqüência ainda maior, pois um conteúdo técnico pode ficar obsoleto com rapidez. *Software* educativos que representam disciplinas sempre em rápida evolução tem que acompanhar esta evolução com a mesma rapidez. Se o time de desenvolvimento deseja que a aplicação educacional possua vida longa, ela não poderá ser escrita de forma fechada, ou seja novas versões devem ser incorporadas com facilidade e certamente sem a necessidade de intervenção de algum suporte técnico, feita pelo próprio usuário, sob pena do usuário descartar o *software* em pouco tempo.

7.2 – Controle de Versões - A manutenção e atualização de *software* educacional requer alguma forma de controle de versão. Um padrão comercial para este controle :

- *Pre-release alpha ou beta* : versões com número menor que 1 (0.5, 0.6, etc);
- *First Release* :Versão 1.0;
- *Correção de Bugs* : Adicionar um letra no número da versão, por exemplo 1.1a;
- Poucos melhoramentos : Adicionar um ponto decimal no maior número de versão para incorporar poucas funcionalidades ou correção de *bugs*, por exemplo 1.1, 1.2;
- Grandes atualizações : Adicionar um número inteiro para representar uma atualização mais substancial, por exemplo 2.0, 3.0.

7.3 – Distribuição - Um estágio que precisa ser considerado no projeto de um *software*, é a sua distribuição. As aplicações multimídia são particularmente vorazes consumidoras de espaço em disco como imagens em vídeo digital ou então áudio digital, o que torna a distribuição em disquetes impraticável, em termos de tempo e custo. Outras possibilidades são [Ril98] :

- Discos Óticos (*Disk-Based Distribution*) – Discos óticos (*magneto-optical disks*) tem uma grande capacidade , acima de 1.3Gb e podem ser gravados repetidamente assim como discos magnéticos. Atualmente, porém, os discos e seus drives são caros e estão instalados apenas em uma pequena quantidade de microcomputadores.

- CDROM – Podem armazenar em média 600Mb e pode ser considerado uma boa opção comercial para distribuição da aplicação. A principal desvantagem da sua utilização é que a produção inicial de cópias ainda é cara.
- Networks – A distribuição através da rede possui um número significativo de vantagens sobre a distribuição por disco (*disk based*) : rapidez, facilidade, redução de custos e flexibilidade para atualizações. Algumas desvantagens a respeito desta forma de distribuição : A necessidade da existência de pontos de rede para a distribuição, a forma de pagamento, a segurança e a documentação se extensa pode vir a ser um custo extra a ser contabilizado.

7.4 – Rotinas de Configuração - Idealmente a instalação deve ser simples. As aplicações comerciais modernas possuem uma instalação fácil. Usuários não devem se preocupar na criação da estrutura de diretórios, ou copiar manualmente arquivos, ou modificar a configuração dos arquivos ou ainda alterar interrupções ou configurações DMA (*Direct Memory Access*), que é uma técnica de entrada/saída de dados do computador por acesso direto à memória [Bia85].

7.5 – Suporte Técnico – O suporte técnico é essencial para o sucesso de um *software* educacional assim como para aplicações em geral. Toda aplicação poderá gerar inúmeros problemas técnicos, particularmente se forem processadas em um PC onde as configurações de *hardware* e *software* tendem a ser das mais variadas possíveis. Os desenvolvedores precisam estar preparados para lidar pacientemente com problemas durante meses ou até anos.

3.0 Conclusão

Ao longo do caminho percorrido para a conclusão deste trabalho, dois importantes aspectos ficaram claramente evidenciados, a preocupação com os objetivos educacionais que deveriam ser atingidos pelo *software* e a utilização de modelos de desenvolvimento para auxiliar na tarefa de construção deste mesmo *software*.

Com relação ao primeiro aspecto constatou-se que a utilização do computador na educação, e por conseguinte o *software*, é vasta e generosa em possibilidades, podendo ser aplicado desde em atividades lúdicas, com ampla liberdade de ação, possibilitando ao aluno como aprender a pensar, passando para sua utilização como mera ferramenta de apoio, chegando até um outro extremo com atividades rigidamente direcionadas pelo *software*.

No tocante ao desenvolvimento do *software* educacional, as pesquisas revelaram uma grande quantidade de ferramentas de autoria, cada qual com notações e abordagens diferentes. É comum, no entanto que essas ferramentas enfatizem determinadas etapas do processo de desenvolvimento, desconsiderando outras. Procuramos, desta forma, apresentar um processo de desenvolvimento, descrevendo suas etapas e sugerindo procedimentos a serem seguidos em cada uma dessas etapas.

Referências Bibliográficas

[Ara97] Araujo, Renata Mendes
Dias, Marcio de Souza
Borges, Marcos Roberto da Silva
Suporte por Computador ao Desenvolvimento Cooperativo de *Software* : Classificação e Propostas
XI Simpósio de Engenharia de *Software* – Ceará – 1997

[Ben97] Bento, Márcia Ferreira
Campos, Maria Luiza
Borges, Marcos Roberto da Silva
Incorporando Funcionalidades de Suporte ao Trabalho Cooperativo Em Ferramentas de Hipertextos.
XI Simpósio de Engenharia de *Software* – Ceará – 1997

[Bia85] Bianchi, Paulo
Bezerra, Milton
Microcomputadores – Arquitetura, Projeto e Programação
LTC - Livros Técnicos e Científicos S.A. - 1985

[Bor97] Borges, Roberto Cabral de Mello
Lima, José Valdeni de
Interface de Navegação em Hiperdocumentos para Aplicações Educacionais
VIII Simpósio de Informática na Educação – São José dos Campos – 1997

[Cys97] Cysneiros, Luis Marcio
Leite, Julio Cesar S. da Prado
Definindo Requisitos Não Funcionais
XI Simpósio de Engenharia de *Software* – Ceará – 1997

[Dra98] Drakos, Nikos
http://cbl.leeds.ac.uk/nikos/tmp/hypermedia/

[Fio98] Fiorini, Soeli
Staa, Arndt Von
Baptista, Renan Martins Baptista
Engenharia de *Software* com CMM
Editora Brasport Livros e Multimídia Ltda – 1998

[Gag85] Gagne, R. M.
The Conditions of Learning and Theory of Instruction
New York – Holt. Rienhart and Wisnton – 1985

[IEE83] IEEE *Standard Glossary of Software Engineering Terminology*
IEEE, ANSI/IEEE Std – New York – 1983.

[Jac92] Jacobson, Ivar.
Christerson, Magnus
Jonsson, Patrik
Övergaard, Gunnar
Object Oriented Software Engineering : A Use Case Driven Approach Ed. Addison Wesley Publishing, 1992".

[Lin95] Lindstron, Robert L.
Guia *Business Week* Para Apresentações em Multimídia
MAKRON *Books* – 1995

[Myl92] Mylopoulos, J. *et alli*
"Representing and Using Non-Functional Requirements : A Process-Oriented Approach"
IEEE Transactions of Software Engennering – Jun/1992

[Ril98] Riley, Fred
Understanding IT : Developing Multimedia Courseware
University of Hull

[Rum94] Rumbaugh, James *et alli*
Modelagem e Projetos Baseados em Objetos
Editora Campus – 1994

[Sta88] Stahl, Marimar M
Avaliação da Qualidade de *Software* Educacional
Relatório Técnico. COPPE/UFRJ. 1988