**UNIVERSIDADE REGIONAL DE BLUMENAU**

**CENTRO DE CIÊNCIAS EXATAS E NATURAIS**

**CURSO DE CIÊNCIAS DA COMPUTAÇÃO**

(Bacharelado)

# PROTÓTIPO DE SOFTWARE EDUCACIONAL PARA ELABORAÇÃO DE EXERCÍCIOS E PROVAS

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

**SUZETE KEINER**

BLUMENAU, JUNHO/2001

2001/1-65

# PROTÓTIPO DE SOFTWARE EDUCACIONAL PARA ELABORAR EXERCÍCIOS E PROVAS

**SUZETE KEINER**

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

**BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO**

_____

Prof.Carlos Eduardo Negrão Bizzotto—Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

**BANCA EXAMINADORA**

_____

Prof. Carlos Eduardo Negrão Bizzotto

_____

Prof. Maurício Capobianco Lopes

_____

Prof. Luiz Bianchi

# AGRACECIMENTOS

À todos que participaram da realização deste trabalho. Especialmente ao meu orientador, Carlos Eduardo Negrão Bizzotto, pela paciência e orientação que me deu.

Ao Professor Marcel Hugo, que participou com sugestões na parte da especificação, que envolveu Orientação a Objetos.

À direção da Escola Municipal “Maurício Germer” por ter disponibilizado todo o material que necessitei na área de educação, para a fundamentação do trabalho.

Valdir, meu namorado, pelos momentos alegres e tristes durante a graduação. Que esteve sempre ao meu lado, me incentivando, impedindo que eu desistisse de lutar.

A Deus e a todos que direta ou indiretamente, ajudaram para tornar o meu sonho uma realidade.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS



# LISTA DE QUADROS

# RESUMO

O presente trabalho apresenta o desenvolvimento de um software de autoria, que permita ao professor a criação de exercícios e provas para seus alunos. De acordo com a metodologia de ensino adotada, o professor irá criar o conjunto de exercícios e provas que melhor se adeqüem à sua realidade. Baseia-se na metodologia OOHDM, utilizada na análise, sendo esta recente e muito utilizada para criação de aplicações hipermídia. Para implementação utilizou-se o software de autoria Macromedia Director 8.0.

# ABSTRACT

The present work presents the development of a responsibility software, that allows to the teacher the creation of exercises and tests for its students. In agreement with the adopted teaching methodology, the teacher will create the group of exercises and tests that better if adapt to its reality. It bases on the methodology OOHDM (Object-Oriented Hypermidia Design Model), used in the analysis, being this recent one and very used for creation of applications hypermidia. For implementation the responsibility software Macromedia Director 8.0 was used.

# 1 INTRODUÇÃO

As novas tecnologias e os avanços na informática nos últimos anos, segundo Andres (2000), vêm oferecendo diversidade de opções para o aprimoramento pessoal e intelectual, que projetam o ser humano no mundo. Os computadores fazem parte da vida dos seres humanos, podendo ser encontrados, nos mais variados locais de trabalho e casas. Sendo assim, softwares são desenvolvidos com a finalidade de acompanhar estas evoluções. Neste ambiente, profissionais ligados ao ensino, nas mais diversas áreas do conhecimento, começam a perceber a importância da utilização dos recursos computacionais como ferramentas de apoio nas disciplinas por eles ministradas.

Atualmente, no mercado brasileiro, existe uma grande quantidade de softwares educacionais para as mais variadas áreas do conhecimento. No entanto, a maioria destes softwares possuem o aluno como público-alvo. Com isso, o professor é alijado do processo, passando a exercer o papel de "selecionador" do software a ser utilizado. Em função disso, o professor, freqüentemente, precisa adaptar seu material didático e/ou sua metodologia ao software escolhido.

Dentro desta realidade, alguns softwares, como o AulaNet, se destacam por serem uma ferramenta para que o professor possa utilizar os recursos da informática para a criação de seu próprio material didático, utilizando sua própria metodologia de ensino-aprendizagem. Estes softwares de auxílio ao professor, conforme alguns conceitos segundo Andres (2000), adotam abordagens distintas de implementação, tais como:

a) exercício e prática: visam a aquisição de habilidade ou aplicação de um conteúdo já conhecido pelo aluno, inteiramente dominado;
b) tutorial: os programas tutoriais podem introduzir conceitos novos, apresentar habilidades, proporcionar aquisições de conceitos, princípios e / ou generalizações;
c) simulação e modelagem: é a representação ou modelagem de um objeto real, de um sistema ou evento, por meio de um modelo simbólico ou representativo da realidade;
d) jogos: os jogos devem ser fonte de recreação com vista a aquisição de um determinado tipo de aprendizagem;

e) hipertexto / hipermídia: hipertexto é definido como uma forma não linear de armazenamento de informações, ou seja, em qualquer ordem, através da seleção de tópicos de interesse, as informações podem ser examinadas;
f) tutores inteligentes: o objetivo dos tutores inteligentes é trazer mais flexibilidade e interatividade no domínio da tutoria, sobretudo em matemática, programação e medicina. Estes sistemas podem ser definidos como uma integração da Inteligência Artificial (IA) e uma teoria da psicologia de aquisição de conhecimento;
g) hiperdocumento no ambiente de redes: através da internet, a *World Wide Web*, também conhecida como WWW, permite ao usuário buscar e recuperar informações distribuídas por diversos computadores que integram a rede e suportam o serviço. Todos eles, podendo ser avaliados e utilizados, conforme objetivos projetados;
h) autoria: são softwares que permitem ao usuário a criação de seus próprios conteúdos, ou seja, o usuário é o autor de seu próprio trabalho.

As aplicações hipermídia constituem importantes ferramentas para tratar informações não estruturadas através de uma interface amigável, as quais têm sido longamente utilizadas em diversas áreas de aplicação. Porém nestas aplicações percebe-se deficiências no que diz respeito ao controle, manutenção e reutilização de um grande volume de informações, já que suas modelagens não seguem os conceitos do paradigma de objetos.

A modelagem *Object Oriented Hypermidia Design Model* (OOHDM), é recente e muito utilizada para desenvolver aplicações hipermídia. Procura fornecer um modelo de estruturação mais natural para as informações, através de classes, que definem a estrutura, e de suas instâncias (nós) que efetivamente mantêm a informação. Desta forma, é possível associar maior semântica às unidades do hiperdocumento, e, por conseqüência, dar suporte a mecanismos mais sofisticados de consulta e organização dos dados, além de facilitar a reutilização e manutenção (Pompermaier, 2000).

Dentro deste contexto, esse trabalho visa o desenvolvimento de um software de autoria, que permita ao professor a criação de exercícios e provas para seus alunos. De acordo com a metodologia de ensino adotada, o professor irá criar o conjunto de exercícios e provas que melhor se adeqüem à sua realidade.

## 1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é o desenvolvimento de um protótipo de software para auxiliar professores de qualquer área do conhecimento, na criação de exercícios e avaliações de aprendizagem a serem realizados pelos seus alunos.

Os objetivos específicos do trabalho são:

a) permitir ao professor criar testes de múltipla escolha, com avaliação automática do desempenho do aluno;
b) possibilitar a criação de perguntas subjetivas, onde os alunos deverão discorrer sobre um dado assunto;
c) permitir a criação de um jogo de quebra-cabeça para que o aluno possa praticar o conteúdo trabalhado.

## 1.2 ESTRUTURA DO TRABALHO

Este trabalho está dividido em oito capítulos. O primeiro capítulo trata dos objetivos do trabalho, suas limitações, justificativa, bem como sua estrutura.

O segundo capítulo, aborda o assunto Educação. Aborda mais especificamente a Informática na Educação. A história da informática na educação e as etapas de implantação da informática educacional. Resultados esperados, assim como vantagens e limitações atuais desta implantação.

O terceiro capítulo trata do Software Educacional. Além de definir o que se entende por software educacional, trata dos aspectos ergonômicos, avaliação de software educacional, qualidade de software educacional e é feita uma análise de alguns dos softwares educacionais existentes.

O quarto capítulo apresenta a metodologia Orientada a Objetos. Neste capítulo encontra-se conceitos básicos da orientação a objetos, benefícios, e é apresentada a Linguagem de Modelagem Unificada (UML). Sendo que esta modelagem, principalmente o diagrama de classes, faz parte da metodologia OOHDM.

O quinto capítulo apresenta a metodologia *Object Oriented Hypermedia Design Method* (OOHDM). Sendo esta, uma metodologia que aborda conceitos de orientação a objetos, estrutura fundamental no desenvolvimento de sistemas.

O sexto capítulo trata do desenvolvimento do protótipo, seguindo os passos da metodologia OOHDM. Neste capítulo são apresentadas as características gerais do protótipo, seu processo de desenvolvimento e algumas telas representativas do protótipo desenvolvidas a partir do Macromedia Director 8, utilizando a linguagem Lingo.

No sétimo capítulo, é apresentada a conclusão do trabalho e algumas sugestões para futuros trabalhos.

# 2 INFORMÁTICA NA EDUCAÇÃO

Um novo milênio chega marcado por mudanças e inovações. Dentro de um processo mundial de globalização, inicia-se um período no qual ter poder significa, principalmente, ter domínio sobre informações – saber acessar, selecionar, aplicar adequadamente as informações necessárias e úteis na construção do saber.

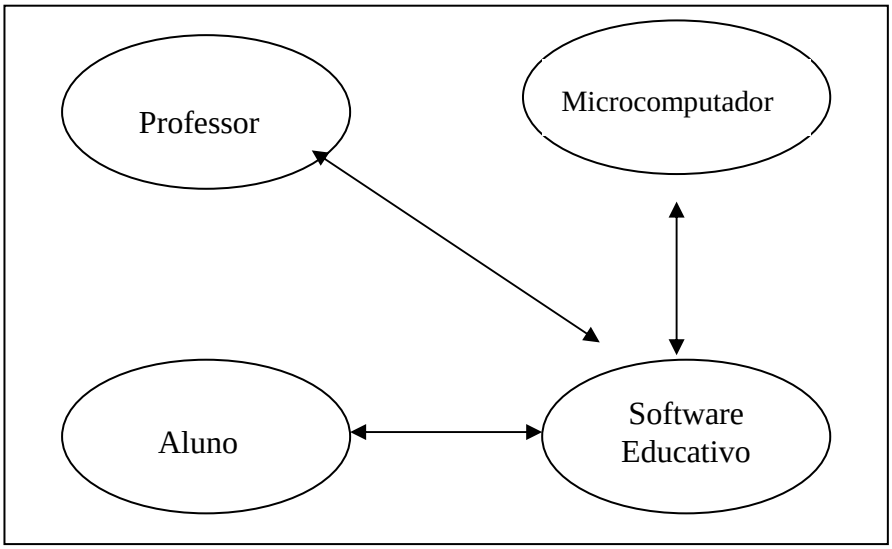
Dominar os fatos e aprender as técnicas de utilização do novo, são atividades importantes, mas que devem ser complementadas com a busca pelo desenvolvimento da capacidade de análise e interpretação desses fatos, na atribuição de um sentido ou significado às inovações. Só assim é possível, além de explicar o novo, apontando suas vantagens e benefícios, antever problemas, propondo soluções criativas na sua resolução.

Não basta apenas saber como funciona um computador, seus componentes de hardware e software e nem comprar todos os softwares educacionais disponíveis. Qualquer proposta de introdução dos computadores na escola, deve considerar a importância e necessidade dos professores conhecerem esse novo instrumento didático, cuja característica fundamental é proporcionar novas formas de ensinar e aprender.

Hoje, no Brasil, vêm sendo desenvolvidas diversas pesquisas relacionadas ao uso do microcomputador em sala de aula e também quanto ao desenvolvimento de software para os mais diversos conteúdos programáticos (Lucena, 2000). O uso do microcomputador pode ter diversas utilidades, como: treinamento profissionalizante, entretenimento e também como ferramenta de ensino dos conteúdos programáticos das diversas disciplinas, contribuindo desta forma, para o aumento da eficiência do ensino.

Os itens envolvidos na implantação da informática na educação, conforme demonstra a Figura 2.1, consiste basicamente no computador, o software educativo, o professor capacitado para usar o computador como ferramenta de ensino e o aluno.

**FIGURA 2.1 – IMPLANTAÇÃO DA INFORMÁTICA NA EDUCAÇÃO**




Em relação ao aluno, o software educativo tem como objetivos: fazer revisões do conteúdo disciplinar já visto em sala de aula (aprimorando seus conhecimentos), avaliar as reais capacidades de aprendizado (analisando erros e avaliando estilos) e despertar o interesse para o aprendizado através da descontração e interatividade (jogos).

O software educativo voltado ao professor, tem por objetivo, segundo Dettmer (1996), treiná-lo, além de demonstrar a teoria, propor exercícios que o professor pode apresentar aos alunos em sala de aula e demonstrar como os conceitos são percebidos pelos alunos de forma geral e o que o professor deve fazer para orientar os alunos na construção dos exercícios.

Segundo Tajra (1998), o uso da informática dentro de um ambiente educacional varia de acordo com a proposta pedagógica que está sendo utilizada em cada caso. É muito importante que as pessoas incorporadas nestes projetos estejam dispostas a novos desafios. As vantagens freqüentemente encontradas são:

a) os alunos ganham autonomia nos trabalhos, podendo desenvolver boa parte das atividades sozinhos, dentro de suas características pessoais;
b) em função da gama de ferramentas disponíveis nos softwares, os alunos, além de ficarem mais motivados, também, tornam-se mais criativos;
c) os ambientes tornam-se mais dinâmicos e ativos, os alunos que sobressaem pelo uso da tecnologia costumam ajudar àqueles que estão com dificuldades;

d) as aulas expositivas perdem espaços para os trabalhos cooperativos;
e) estímulo a uma forma de comunicação voltada para a realidade atual de globalização;
f) a maioria dos programas disponíveis no mercado são em outros idiomas, sendo que esta característica em si não deve ser vista como empecilho, mas como uma motivação para o aprendizado de novos idiomas;
g) além da escola poder direcionar as fontes de pesquisas para os recursos já existentes, tais como livros, enciclopédias, revistas, jornais e vídeos, ela pode optar por mais uma fonte de aprendizagem : o computador;
h) uma grande contribuição da informática é o auxílio para o desenvolvimento das habilidades de comunicação e de estrutura lógica do pensamento.

Contudo, apesar dessa aparente evolução, pode-se considerar que a educação ainda está precária quanto a utilização de computadores como ferramenta para a aprendizagem. Parte das dificuldades para a utilização de recursos de informática na educação e a resistência dos professores em se adequar às novas tecnologias, está ligada às tentativas de implantação da informática nas atividades escolares (Tajra, 1998).

O livro foi um dos primeiros instrumentos tecnológicos inclusos no processo de ensino-aprendizagem, o que causou na época muitas alterações educacionais, estando hoje totalmente incorporado. A implantação da informática na área da educação ainda é recente, pois muitos questionam os efeitos de sua utilização, mas não vêem possibilidades de não utilizá-los.

O importante, ao se utilizar dos recursos tecnológicos à disposição das práticas pedagógicas, é questionar o objetivo que se quer atingir, avaliando sempre as virtudes e limitações de tais recursos.

## 2.1 HISTÓRIA DA INFORMÁTICA NA EDUCAÇÃO

O uso da tecnologia de comunicação e informação na educação tem crescido nas últimas décadas, principalmente, a partir do advento da microinformática e, mais recentemente, com a expansão da Internet. Segundo relatório do Departamento de Educação dos Estados Unidos (*US Departament of Education*, 1997), a partir de 1994 tem havido um crescimento anual de 15 % no número de escolas públicas com acesso à Internet. Este crescimento ocorre em todas as regiões do país, demonstrando um grande interesse do governo americano em "conectar todas as escolas públicas à Internet". Segundo este relatório, a tendência é de que no ano 2001, mais de 95% das escolas públicas dos Estados Unidos estejam ligadas à Internet (in Bizzotto, 1999).

Na Inglaterra, de acordo com relatório do Departamento para Educação e Emprego (*Departament for Education and Employment*, 1998), foram gastos 174,5 milhões de libras na aquisição de computadores e softwares para as escolas. A Tabela 2.1.1 mostra os principais dados deste relatório.

**TABELA 2.1.1 – TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO NA INGLATERRA**

| | Primário | Secundário | Especial |
|---|---|---|---|
| Número médio de computadores por escola | 13 | 101 | 19 |
| Média do número de alunos por computador | 18 | 9 | 4 |
| Gastos em tecnologia da informação e comunicação para o ensino e aprendizagem (milhões de libras) | 48,5 | 118,6 | 7,4 |
| Percentual de escolas conectadas à Internet | 17% | 83% | 65% |

FONTE : in Bizzotto (1999)

De acordo com a tabela 2.1.1, o investimento em computadores e softwares do governo inglês tem se concentrado no ensino secundário, diferentemente do que ocorre nos Estado Unidos onde o investimento é equivalente tanto no nível elementar quanto no nível secundário.

Em 1983, 53% das escolas dos EUA, já utilizavam computadores com grande apoio de empresas privadas que atuavam nesta área. A França, por meio do Plano "Informática para

Todos", e a Espanha, por meio do Projeto *Atenea*, estimularam a formação de professores para o atendimento de vários alunos. Na área de informática educacional não existe um modelo universal para a sua aplicação, ela varia de acordo com a disponibilidade de recursos humanos, financeiros e técnicos, das linhas metodológicas das escolas, bem como, da própria credibilidade em relação à tecnologia na educação (Tajra, 1998).

O governo brasileiro iniciou várias ações no sentido de instalar computadores na área educacional de 1° e 2° graus da rede pública, visando, assim como outros países, melhoria da qualidade das escolas, de tal forma que fosse possível garantir aos alunos o acesso ao conhecimento de uma tecnologia bastante utilizada na sociedade moderna.

Formalmente, a informática na educação foi introduzida no Brasil pela Universidades de Campinas, Estado de São Paulo (UNICAMP), em 1973. Naquele ano a UNICAMP criou um grupo de pesquisa interdisciplinar nas áreas de computação, lingüística e psicologia educacional do qual resultou a elaboração do Projeto LOGO ( MEC, 2001).

O Projeto LOGO foi operacionalizado a partir de 1978, ano em que, de fato, a informática chegou à escola pública brasileira, em Campinas. O interesse despertado pelo Projeto LOGO fez com que a UNICAMP criasse em 1983, o Núcleo Interdisciplinar de Informática Aplicada à Educação – NIED.

Data também, de 1984, o lançamento de Projeto EDUCOM, a primeira iniciativa em larga escala do governo brasileiro na área de informática na educação. Este projeto teve como objetivo geral "fomentar o desenvolvimento da pesquisa multidisciplinar e a formação de recursos humanos voltados para a aplicação das tecnologias de informática na educação".

O Projeto EDUCOM originou o Programa Nacional de Informática Educativa – PRONINFE, lançado em 1989, para apoiar o desenvolvimento e a utilização das novas tecnologias de informática no ensino fundamental, médio e superior e na educação especial. Tanto o EDUCOM quanto o PRONINFE não chegaram à escola de ensino básico, permaneceram no campo experimental em universidades, secretarias de educação e escolas técnicas ( MEC, 2001).

Surgiu em 1995, o PROINFO. O Proinfo é desenvolvido pelo Ministério da Educação – MEC ( pela Secretaria de Educação a Distância – SEED) em parceria com os governos

estaduais, visa a introdução das Novas Tecnologias de Informação e Comunicação (NTIC) na escola pública, como ferramenta de apoio ao processo de ensino-aprendizagem. É portanto, um programa de educação. O Proinfo tem diretrizes estabelecidas pelo MEC e pelo CONSED ( Conselho Nacional de Secretários Estaduais de Educação). Em cada unidade da federação há uma Coordenação Estadual de Informática na educação, cujo presidente é nomeado pelo Secretário de Educação, composta, basicamente, por representantes desta secretaria, universidades, algumas secretarias municipais de educação e da comunidade escolar (pais, professores, diretores, etc).

O Proinfo tem a preparação de recursos humanos, professores, especialmente, como a principal condição de sucesso. Professores são preparados em dois níveis: multiplicadores e de escolas (MEC, 2001).

Um professor-multiplicador é um especialista em capacitação de professores (de escolas) para uso da telamática em sala de aula : adota-se no Programa, portanto, o princípio professores capacitando professores. É formado por cursos de pós-graduação (especialização *lato sensu*) ministrados por universidades brasileiras (públicas ou privadas, escolhidas em função da excelência na área do uso de tecnologia na educação).

Os multiplicadores capacitam os professores das escolas nos Núcleos de Tecnologia Educacional – NTE. Um NTE é uma estrutura descentralizada do Proinfo, especializada na área de telemática aplicada à educação (MEC, 2001):

a) capacitação de Recursos Humanos (professores e técnicos de suporte);
b) suporte pedagógico e técnico a escolas (elaboração de projetos de uso pedagógico da telemática e respectivo acompanhamento, suporte a professores e técnicos, etc);
c) pesquisas.

Os objetivos do Programa Nacional de Informática na Educação (PROINFO) são:

a) melhorar a qualidade do processo de ensino e aprendizagem;
b) possibilitar a criação de uma nova ecologia cognitiva nos ambientes escolares, mediante incorporação adequada das novas tecnologias de informação pelas escolas;
c) propiciar uma educação voltada para o desenvolvimento científico e tecnológico;
d) educar para uma cidadania global numa sociedade tecnologicamente desenvolvida.

O Programa está sendo implantado em todos os estados do Território nacional, e a distribuição dos computadores será de acordo com o número de alunos matriculados em cada estado.

O Proinfo adquirirá em sua primeira etapa (até 2002) cerca de 105.000 computadores, que serão instalados nos NTE e nas escolas (6000) em todo o país, beneficiando cerca de 7.5 milhões de alunos. Cada unidade da federação tem uma quota percentual definida proporcionalmente ao número de alunos e escolas de sua rede pública de ensino. O mesmo critério foi aplicado para determinação do número de NTE por UF, considerando 250 o total de NTE no Brasil (MEC, 2001).

As diretrizes do Programa prevêem que só receberão computadores (e respectivos periféricos) escolas que tenham um projeto de uso pedagógico da telemática aprovado pelas respectivas coordenações estaduais e, além disso, disponham de:

e) recursos humanos capacitados para implementar tal projeto;
f) ambiente adequado para instalação de equipamentos ( que tenha segurança, alimentação elétrica de qualidade e um mínimo de conforto para alunos e professores).

As escolas são vistoriadas antes do envio dos equipamentos e há um sistema informatizado de acompanhamento do processo de instalação de equipamentos nas escolas e nos NTE. Atualmente, está em desenvolvimento um sistema informatizado (tecnologia WEB) de avaliação do Programa (avaliação finalística, voltada para determinar como a introdução da telemática na rede pública de ensino influi na formação do aluno, na qualidade da escola).

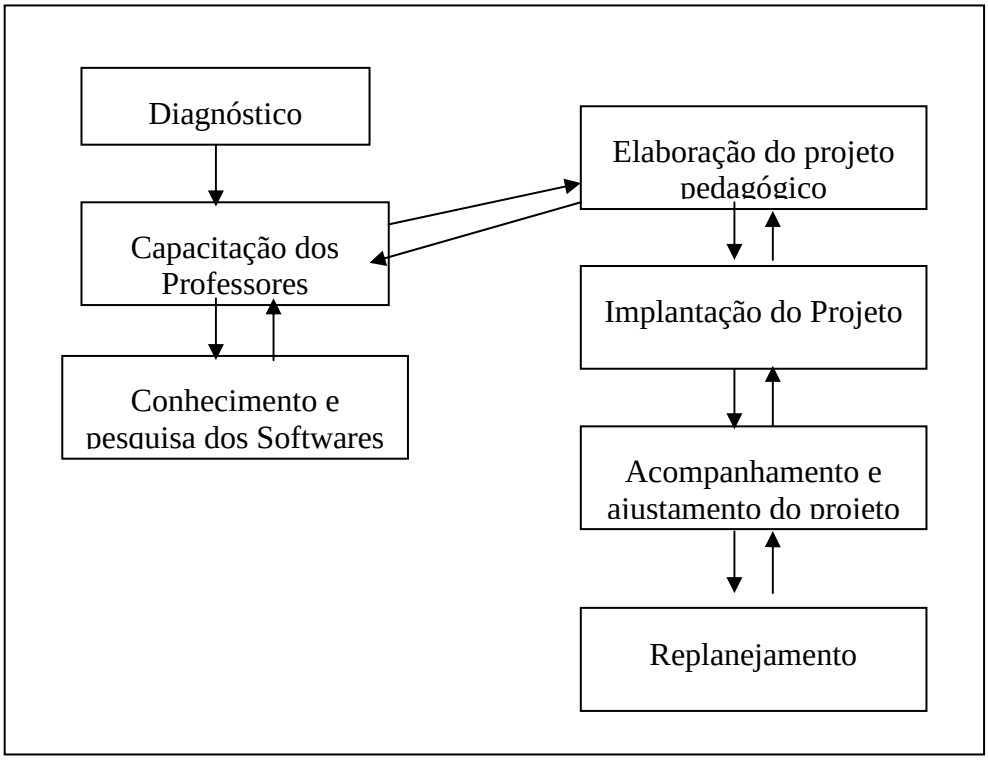
Como parte importante da estratégia de consolidação do Proinfo, foi instalado o Centro de Experimentação em Tecnologia Educacional – CETE, concebido para apoiar o processo de incorporação de tecnologia educacional pelas escolas e para ser um centro de difusão (e discussão, em rede) de experiências e conhecimentos sobre novas tecnologias aplicáveis à educação. O CETE é , também, o elemento de contato brasileiro em iniciativas internacionais vinculadas à tecnologia educacional e à educação a distância.

Independente de participar do Proinfo, uma escola deve analisar com cuidado a implantação da Informática na Educação. Neste sentido, torna-se importante planejar as etapas de implantação a serem realizadas (MEC, 2001).

## 2.2 ETAPAS DE IMPLANTAÇÃO DE INFORMÁTICA EDUCACIONAL

Existem diversas formas de se montar um projeto de informática educativa. Segundo Tajra (1998), as etapas para a implementação desse projeto podem ser divididas da seguinte forma, conforme mostra a figura 2.2.1 :

**FIGURA 2.2.1 – ETAPAS DE IMPLANTAÇÃO DE INFORMÁTICA EDUCACIONAL**



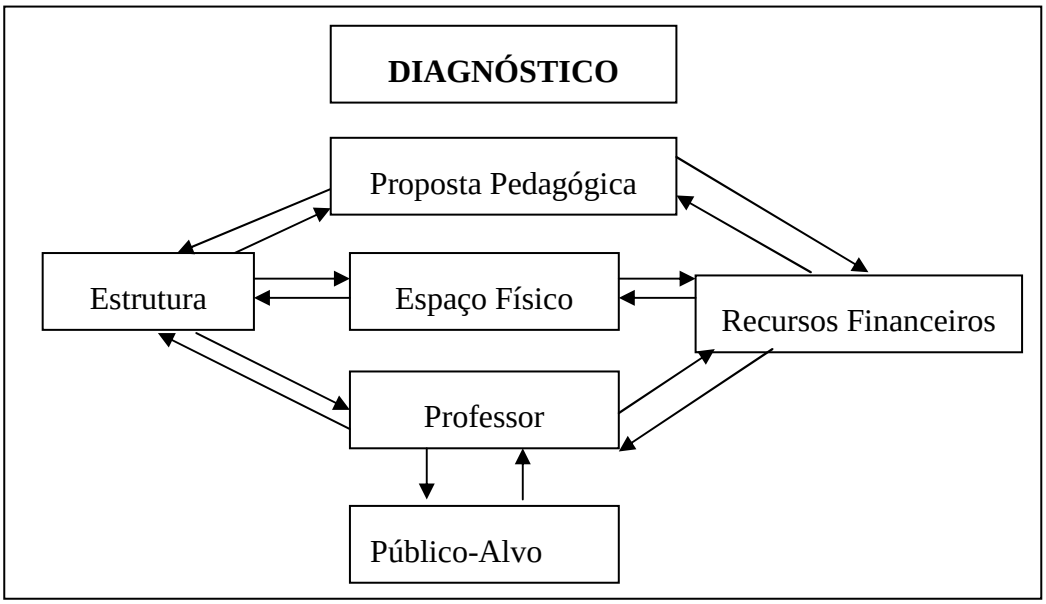

a) diagnóstico: nesta etapa a escola deverá analisar diversos itens, demonstrados na figura 2.2.2:

- proposta pedagógica: em geral pode-se citar três modalidades. Informática como fim em si mesma, informática relacionada a softwares baseados em enfoques disciplinares ou integrar a utilização da informática no projeto educacional da

escola como um todo, tomando para si o tema gerador definido, a ser trabalhado pela escola;

- recursos financeiros: a escola deve, inicialmente, fazer um levantamento do números de alunos por sala e, a partir deste número, prever a quantidade de computadores a ser adquirida;
- estrutura: é importante avaliar se os computadores devem ser interligados em rede, para que se possa projetar a sala conforme recursos computacionais exigidos. Deve-se providenciar o compartilhamento de softwares e equipamentos (como a impressora);
- espaço físico: definição do espaço físico que poderá ser disponibilizado para o ambiente de informática, levando em consideração aspectos como: iluminação, temperatura recomendada para os computadores, *layout* para facilitar o gerenciamento das máquinas, para o caso de manutenção, por exemplo;
- público-alvo: identificar séries (classes) que farão uso do computador;
- professores: definir quais os professores envolvidos e que deverão ser capacitados.

**FIGURA 2.2.2 – DIAGNÓSTICO DO PROJETO DE INFORMÁTICA EDUCATIVA**



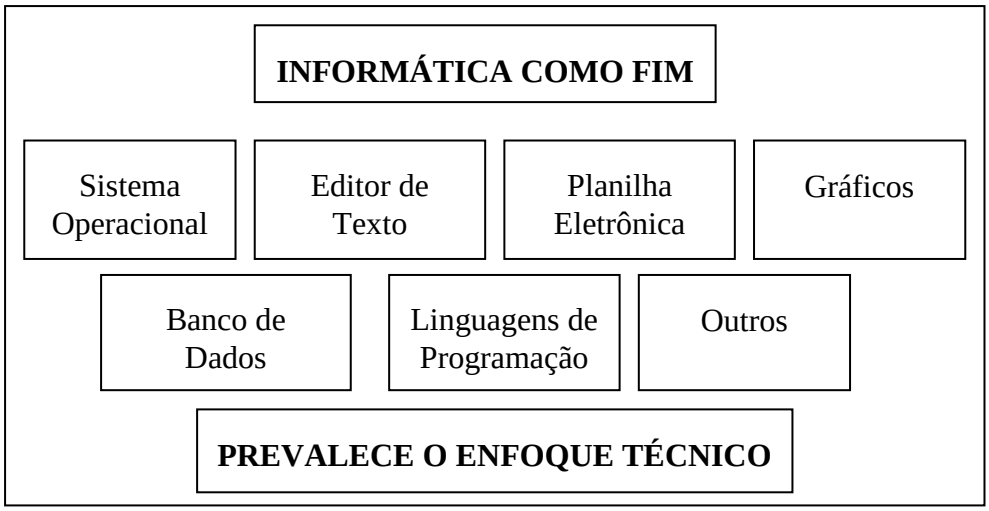
b) capacitação dos professores: nesta etapa, a escola deverá prover condições para que os docentes possam ser capacitados nos aspectos que irão afetar diretamente a implantação da informática na área educacional.

c) conhecimento e pesquisa dos softwares: definição dos softwares que serão utilizados, conforme a modalidade de utilização dessa tecnologia optada pela escola e de acordo com a escolha dos professores envolvidos.

d) elaboração do Projeto Pedagógico com o uso da informática educativa: após a capacitação dos professores, deverá ser definida a linha mestra da informática dentro da escola, conforme as três modalidades :

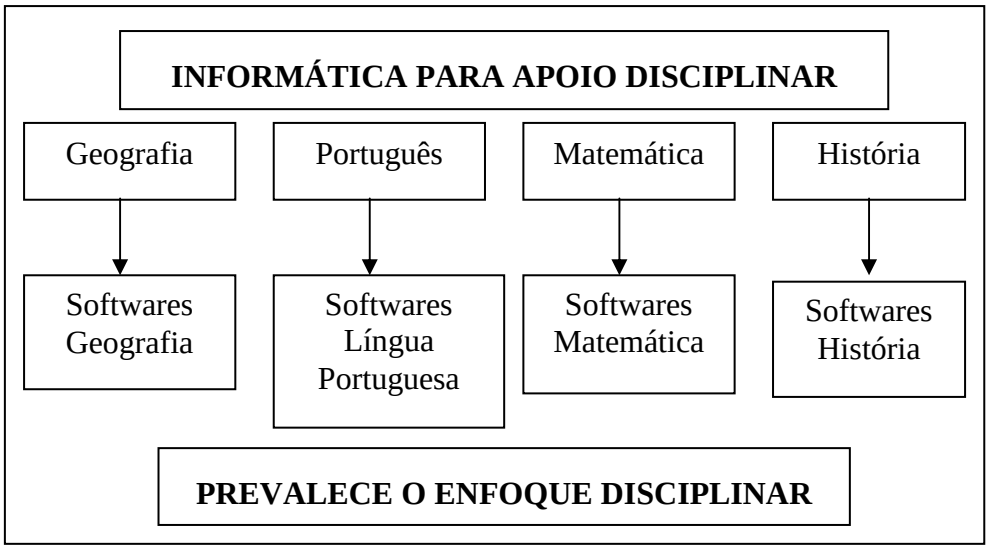
- informática como um fim: que baseia-se no estudo das ferramentas disponíveis nos programas, sem nenhuma relação com os assuntos e temas estudados na escola. A figura 2.2.3 mostra exemplos de algumas destas ferramentas.

**FIGURA 2.2.3 – FERRAMENTAS DISPONÍVEIS NOS PROGRAMAS**



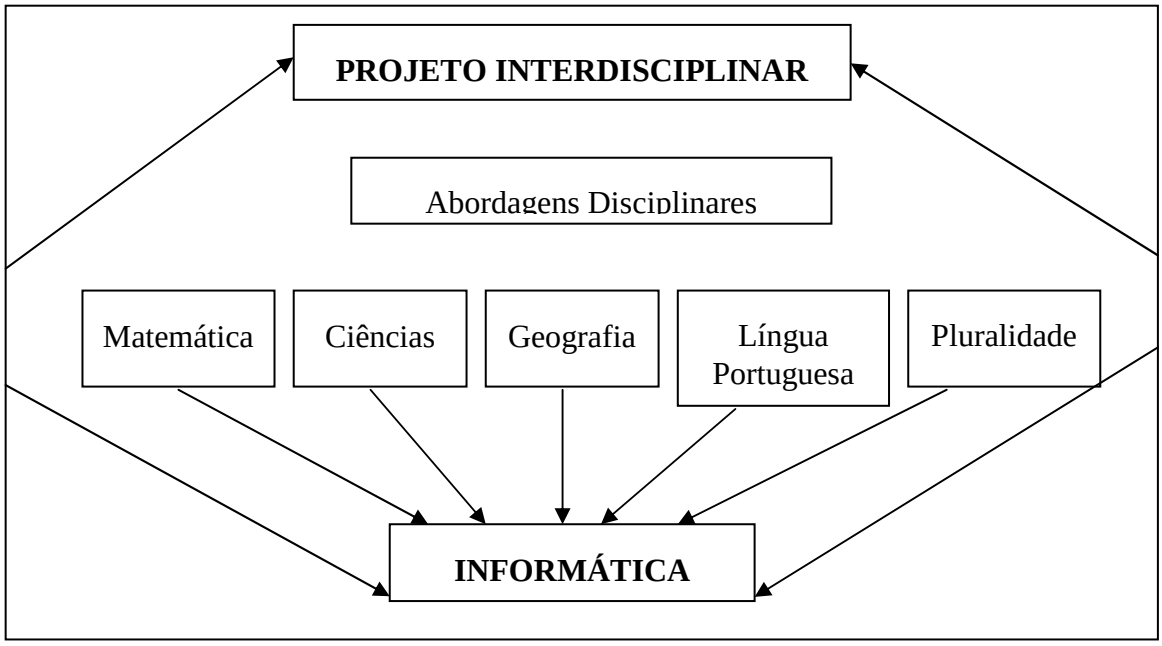

- Informática como apoio: neste caso, são utilizados, de forma isolada, softwares educacionais para as disciplinas existentes, sem relação com os trabalhos disciplinares. Procura-se nestes softwares apoio no processo de aprendizagem da disciplina em contexto. A figura 2.2.4 demonstra um exemplo deste processo.

**FIGURA 2.2.4 – DISCIPLINAS EXISTENTES NOS SOFTWARES EDUCACIONAIS**




- Informática por meio de projetos disciplinares : uma das propostas atuais do ensino está voltada para a interdisciplinaridade através da realização de projetos didáticos. Um projeto didático surge de um “Tema Gerador” sugerido pelos alunos, a partir do qual os professores elaboram os objetivos específicos na sua disciplina em consonância com o objetivo geral para o projeto. A construção de um projeto deve considerar determinados aspectos para que haja unidade de propósitos, consistência nas ações, sentido comum nos esforços de cada um e resultados sistematizados. Conforme exemplo de Tajra, (1998), e como mostra a figura 2.2.5, foi escolhido o tema gerador “Doenças Sexualmente Transmissíveis, que apresenta como objetivo, despertar nos alunos as principais formas de evitar as doenças sexualmente transmissíveis, bem como, os danos que estas podem causar em nossa saúde.

**FIGURA 2.2.5 – INFORMÁTICA POR MEIO DE PROJETOS INTERDISCIPLINARES**




e) implantação do projeto: é a etapa da execução das atividades previamente planejadas, onde os professores freqüentam o ambiente de informática educativa e vivem o desafio a que eles se lançaram.

f) acompanhamento e ajustamento do projeto: etapa em que todos os envolvidos no processo farão a avaliação dos resultados das aplicações, promovendo os devidos reajustes.

g) replanejamento: etapa em que, diante do que foi implantado e avaliado, há a necessidade de que a equipe reveja todos os pontos fortes e fracos do processo, para que as próximas ações não se deparem com as mesmas situações negativas anteriormente vividas.

# 3 SOFTWARE EDUCACIONAL

O computador pode ser um importante recurso para promover a passagem da informação para o usuário ou promover a aprendizagem. No entanto, da análise dos softwares é possível entender que o aprender não deve estar restrito ao software, mas à interação professor-aluno-software. Alguns softwares apresentam características que favorecem a atuação do professor, como no caso da programação; outros em que certas características não estão presentes e requerem um maior envolvimento do professor para auxiliar o aluno a aprender, como no caso do tutorial (MEC, 2001).

Assim a análise dos softwares educacionais em termos do aprendizado e do papel que o professor deve desempenhar para que o aprendizado ocorra, permite classificá-los em posições intermediárias entre os tutoriais e a programação. No entanto, cada um dos diferentes softwares usados na educação, apresentam características que podem favorecer, de maneira mais explícita, o processo de construção do conhecimento. É isso que deve ser analisado quando se escolhe um software para ser usado em situações educacionais.

No entanto, é necessário entender que qualquer tentativa para analisar os diferentes usos do computador na educação é problemática e pode resultar em uma visão muito simplista sobre o software e seu uso. Porém, pode ser um exercício interessante  e ajudar e entender o papel do computador e como ele pode ser efetivo no processo de construção do conhecimento (MEC, 2001).

## 3.1 ASPECTOS ERGONÔMICOS

Para que a interface de um sistema educacional seja bem sucedida, deve ser coerente na determinação da aparência e do comportamento. Grande parte das ferramentas utilizadas para desenvolvimento de software educacionais, que utilizam recursos de multimídia, oferecem ferramentas para desenhar e desenvolver uma interface gráfica bastante amigável ao usuário.

Os produtos de software educacional procuram imitar o que acontece na escola. As razões deste tipo de aplicativo ser prevalecente nas escolas estão relacionadas com a falta de experiência didático-pedagógica por parte dos projetistas destes softwares, implementando muitas vezes teorias pedagógicas ultrapassadas (Campos, 1994).

Sendo assim, a qualidade educacional do produto de software a ser desenvolvido e/ou adquirido, vai depender de uma variedade de situações de aprendizagem que ele propiciar.

## 3.2 AVALIAÇÃO ERGONÔMICA DE SOFTWARE EDUCACIONAL

A Ergonomia é uma ciência que busca a adaptação do ambiente técnico e organizacional ao homem, com a finalidade de obter a satisfação e produtividade no trabalho. (Campos, 1994).

Dentro da ergonomia existem estudos sobre a Interface Homem-Computador (IHC), onde são oferecidas bases teóricas e metodológicas capazes de encontrarem as dificuldades relacionadas com o homem e a máquina. Essa ergonomia de IHC pode ser aplicada a qualquer dispositivo de interação e sua conseqüente relação com o grau de satisfação do usuário irá determinar a qualidade ergonômica do dispositivo.

Os objetivos da avaliação ergonômica podem ser:

a) avaliar as funcionalidades (necessidades dos usuários);
b) avaliar o efeito da interface sobre o usuário, que se traduz na facilidade de aprendizagem do software e na eficiência de uso.

A avaliação dos efeitos do software sobre os usuários é um trabalho que pode ser realizado sem a presença direta deles. No entanto, estas técnicas exigem uma carga alta de experiência do avaliador, o qual é treinado para verificar uma série de recomendações ergonômicas (Dettmer, 1996).

Mas quando se deseja uma avaliação mais precisa de um software em questão, ou seja, das suas funcionalidades, faz-se necessário a presença dos usuários alvos para aplicação de questionários, entrevistas e simulações de utilização do software.

## 3.3 QUALIDADE DE SOFTWARE EDUCACIONAL

O tema qualidade de software é motivo de preocupação para todos os desenvolvedores. Qualidade é um conceito multidimensional que envolve tanto as pessoas que desenvolvem software, quanto aquelas que farão uso do mesmo. Pode-se definir qualidade de software

como o conjunto de propriedades a serem satisfeitas em um determinado grau, de modo que o software satisfaça as necessidades de seus usuários. Entretanto, qualidade não pode ser definida universalmente, ou seja, deve ser definida para um item/elemento específico. Para cada produto de software existe um conjunto de características de qualidade mais adequado. Esse conjunto dependerá da natureza e do uso que se espera fazer do produto (Valente,1995).

A avaliação da qualidade de um software educacional deve levar em conta, principalmente, as características relacionadas à qualidade didático-pedagógica. Neste aspecto, os objetivos dos estudiosos da ergonomia de software e dos educadores seguem um mesmo objetivo, garantir a adaptação do trabalho ao homem e aos meios didáticos, a fim de obter a satisfação e produtividade dos alunos no processo de ensino-aprendizagem.

Um software educacional possui características que os diferencia dos demais softwares utilizados no mercado, pois ele deve estar inserido num contexto pedagógico, de aprendizado pré-definido, proporcionar autonomia, cooperação, criatividade, pensamento crítico, descoberta e construção do conhecimento. Desta forma, além de ser intuitivo, fácil de usar e eficiente, o mesmo deve ser didático (Campos, 1994).

## 3.4 TÉCNICAS DE AVALIAÇÃO DE SOFTWARES EDUCACIONAIS

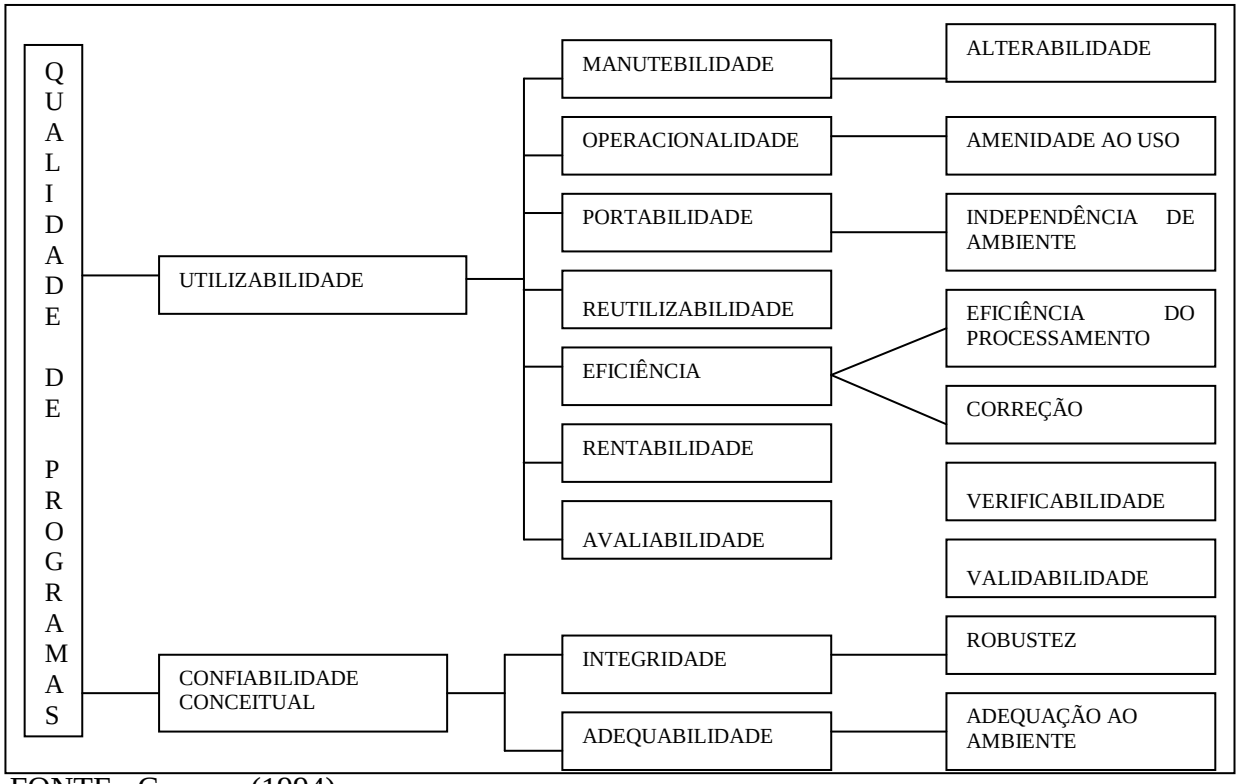
Existem diversas técnicas de avaliação de softwares educacionais, sendo que cada uma delas apresenta abordagens e critérios conforme a necessidade que cada software educacional apresentar para se modelar a qualidade exigida para o público ao qual é concebido. Essas técnicas avaliam, de modo geral, a eficácia global do software, utilizando técnicas que são necessárias para a coleta de dados: observação das reações do usuário-alvo, avaliação das aquisições e impressões sobre a qualidade do software, questionários, entrevistas e ainda ouvir um grupo de especialistas antes de fazer funcionar o software. Muitas utilizam-se de *checklists*, ensaios de interação e avaliações heurísticas, detectam aspectos computacionais e principalmente pedagógicos, sempre inspecionando aspectos ergonômicos dos softwares. (Andres, 2000).

Conforme a técnica de avaliação de *Mucchielli*, citado em Campos (1994), considera-se dez passos a serem examinados na avaliação pedagógica de um software educacional:

a) avaliação das aquisições permitidas, concernentes aos elementos de conhecimento retido ou a medida das *performances* evolutivas, resultado dos teste de avaliações;
b) qualidade do modelo pedagógico adotado;
c) qualidade da idéia geral do software;
d) qualidade e variedades dos procedimentos de interatividade utilizadas;
e) qualidade da flexibilidade do software;
f) natureza e qualidade das ajudas;
g) grau de flexibilidade do software;
h) qualidade das telas;
i) qualidade do documento de acompanhamento;
j) qualidade contínua do produto.

Uma das técnicas muito utilizadas, é o modelo proposto por Campos (1994), que baseia-se nos objetivos, fatores, subfatores, critérios, processos de avaliação, medidas e medidas agregadas, conforme mostra a figura 3.4.1. Caracteriza-se por ser um *checklist*, pois apresenta uma lista de perguntas (critérios) a serem avaliados, bem como o uso da avaliação heurística com o intuito de se fazer o julgamento com relação ao software.

**FIGURA 3.4.1 - OBJETIVOS, FATORES E SUBFATORES DE QUALIDADE**




FONTE : Campos (1994).

Um software deve ser desenvolvido para atender as necessidades do usuário, deve ter vida longa, útil e produtiva. Para isso ele deve atender alguns objetivos, conforme os descritos abaixo ( Campos, 1994):

a) utilizabilidade: determina a conveniência e a viabilidade de utilização do produto ao longo de sua vida útil.
b) confiabilidade conceitual: o produto necessita satisfazer às necessidades e requisitos que motivam sua construção;

A utilizabilidade é um objetivo fundamental para a qualidade de um produto de software e considera as diferentes formas de sua utilização, durante as fases de desenvolvimento e uso. Este objetivo exige a existência da confiabilidade conceitual. Diversos fatores estão relacionados a esse objetivo:

a) manutenibilidade: avalia a facilidade com que o programa pode ser adaptado a fim de atender as necessidades de modificação que surgem depois do seu desenvolvimento;
b) operacionalidade: avalia a facilidade de comunicação com o usuário. Este fator possui dois subfatores: oportunidade e amenidade ao uso;
c) portabilidade: característica de um programa poder ser operado de maneira fácil e adequada em diferentes configurações de equipamento além do original;
d) reutilizabilidade: característica que avalia a possibilidade do reaproveitamento, total ou parcial, de funções desenvolvidas em um programa em outras aplicações;
e) eficiência: característica de o programa realizar suas funções sem desperdício de recursos (memória, periféricos, outros);
f) rentabilidade: característica de o programa ter uma relação custo-benefício aceitável;
g) avaliabilidade: característica que avalia a facilidade com que o programa pode ser avaliado. Este possui dois subfatores: verificabilidade e validabilidade.

O objetivo da Confiabilidade Conceitual, segundo Campos (1994), refere-se às características que garantem que um produto de software atende a seus requisitos, no que se refere ao seu conteúdo. Portanto, avaliar e garantir a confiabilidade conceitual está relacionado a avaliar e garantir a confiabilidade do conteúdo das informações do sistema e propiciar a facilidade de adequação ao ambiente.

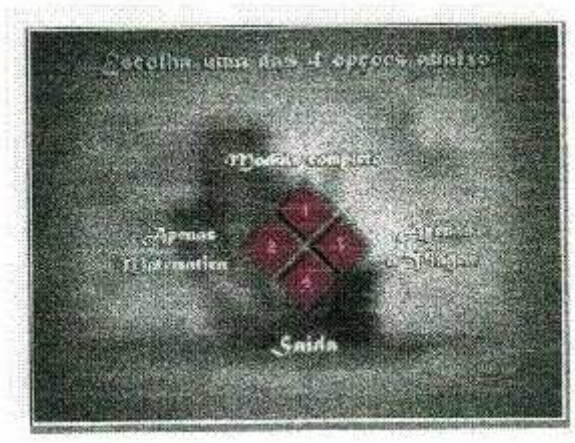
## 3.5 SOFTWARES EDUCACIONAIS AVALIADOS

Conforme ressaltado anteriormente, os softwares educacionais devem estar fundamentados em princípios pedagógicos, independente de quais sejam estes princípios. Existem atualmente no mercado, inúmeros softwares educacionais que são analisados segundo suas características básicas e com relação aos princípios que os fundamenta. A seguir, será realizada uma análise de alguns softwares existentes no mercado, com relação a seus objetivos e estratégia de abordagem (Dettmer, 1996).

a) Software de Exercício e Prática:

- **Trigonometria** : é um software de exercício e prática que possibilita interatividade por meio de respostas às questões apresentadas. Neste programa, o aluno pode optar por quatro modalidades, incluindo o estudo da matemática de uma forma bastante interessante. No módulo “Apenas a Viagem” ele relata a viagem de Colombo, e o software utiliza-se dessa viagem para suas representações matemáticas. Esta junção de geografia, história, matemática, língua portuguesa e, de acordo com o conhecimento do professor, também a física, promove a interdisciplinaridade. A figura 3.5.1 mostra a tela inicial do software Trigonometria.

**FIGURA 3.5.1 – TELA INICIAL DE TRIGONOMETRIA**

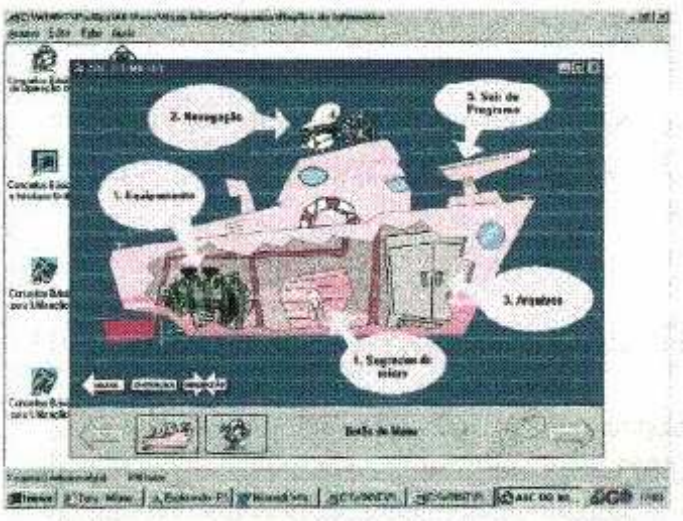



b) Software tutorial e hipertexto:

- **ABC do Micro:** possui um mascote que auxilia o usuário enquanto a navegação é feita pelo software. Este tutorial permite ao usuário o avanço pela

seta ao final da tela, passando por todos os pontos vistos no conteúdo do software ou clicando no item menu, escolhendo que tipo de assunto prefere acessar, conforme mostra figura 3.5.2.

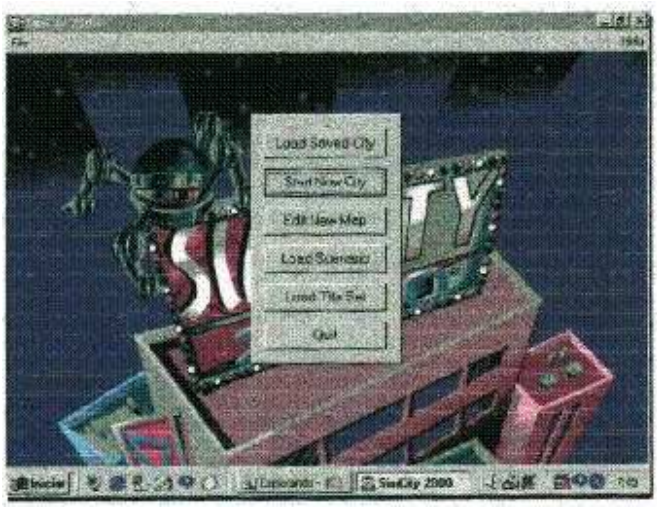
**FIGURA 3.5.2 - TELA DO ABC DO MICRO**



c) Simulação e Modelagem:

- **SimCity 2000:** permite a construção e a administração de uma cidade. Este jogo de simulação leva em consideração as verbas disponíveis, os impostos, as necessidades da população e eventuais desastres que podem afetar a cidade, tais como, acidente nuclear, terremotos, enchentes, incêndios, entre outros. Ao ocorrer um desastre, as medidas de recuperação de uma área devem ser fornecidas e o tempo de recuperação e mudanças são levadas com conta. O uso deste software, no contexto educacional, permite a criação de situações onde alunos e professores podem discutir e propor soluções viáveis para problemas como poluição, trânsito preservação do meio ambiente. A figura 3.5.3 mostra a tela de menu do SimCity 2000(Dettmer, 1996).

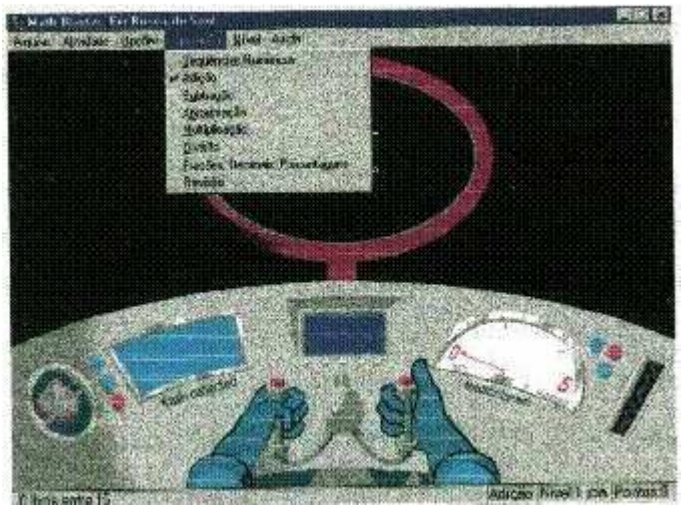
**FIGURA 3.5.3 - TELA DE MENU DO SIMCITY 2000.**




d) Jogos:

- **Math Blaster:** é um jogo educativo. Este jogo permite ao aluno trabalhar com conteúdos da matemática em diferentes assuntos e níveis de dificuldades. A figura 3.5.4, mostra a abertura de menus para seleção dos assuntos. Os desafios na forma de jogos, aumentam a pontuação a medida que o aluno demonstra habilidades nos cálculos propostos para cada fase. Existem jogos educativos para as mais variadas áreas, cabendo aos professores a escolha adequada destes softwares para fixação de conhecimentos vistos em salde aula.

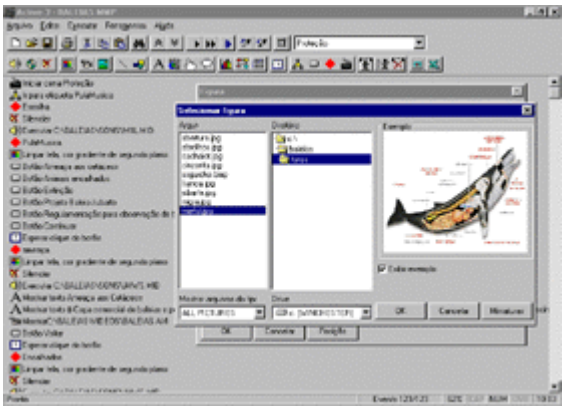
**FIGURA 3.5.4 - TELA DO MATH BLASTER.**



e) Software de Autoria

- **ACTIVE 3:** este software de autoria possibilita a criação de apresentações e aplicações multimídia de modo totalmente amigável e interativo. Através da utilização de um vasto número de ferramentas, mostradas na figura 3.5.5, é possível combinar texto, som e imagem para criar apresentações interativas. Além desses recursos, o ACTIVE 3 oferece um pacote de ferramentas para confecção de fluxogramas, criação de animações e captura de quadros de vídeo. Possui ainda um programa de banco de dados que permite organizar os arquivos que serão utilizados na apresentação. É uma base de dados aberta para administrar e organizar objetos multimídia (textos, imagens, sons, vídeos e animações) conforme o tema, de maneira hierárquica (Dettmer, 1996).

**FIGURA 3.5.5 – TELA PRINCIPAL DO ACTIVE 3**



# 4 METODOLOGIA ORIENTADA A OBJETOS

Uma série de linguagens de programação tem contribuído para a evolução das atuais linguagens orientadas ao objeto, a começar pela LISP, nos anos 50. Esta linguagem, um acrônimo para processamento de relatórios (*List processing*), é uma linguagem de inteligência artificial que introduziu o conceito de ligação dinâmica e o benefício de um ambiente de desenvolvimento interativo para a evolução das linguagens orientadas ao objeto. A Simula, desenvolvida em 1966 como uma linguagem para programação de simulações, foi responsável pelos conceitos de mecanismos de classes, hereditariedade e encapsulamento (Junior,1999).

Com o surgimento de C, nos anos 80, como uma linguagem extremamente popular para programação em todas as plataformas, as suas extensões de orientação ao objeto têm sido uma razão básica para o próprio aumento de atenção à programação voltada ao objeto, e argumento para um maior uso de C++ na comunidade de programação. A partir desta década, surgiram os primeiros estudos sobre o uso da OO para especificação de projetos de sistemas e não foi difícil a utilização dos mesmos conceitos e mecanismos para análise de sistemas.

As aplicações rumam para um processamento maior de conhecimento e será cada vez mais comum a visão de  sistemas utilizando vídeo, imagem, áudio e gráficos. As linguagens orientadas a objeto representam um novo patamar em relação às linguagens procedurais e, mesmo quando suportadas por algumas extensões de linguagens tradicionais como o Pascal e o C, contém características próprias como objetos, classes, mensagens, operações e hereditariedade (Furlan, 1998).

Segundo Furlan (1998), de uma maneira geral, uma linguagem deve possuir quatro elementos de modo a suportar a programação orientada a objeto:

a) proteção de dados: para assegurar a confiabilidade e capacidade de modificação de um sistema pela redução de interdependências entre seus componentes;
b) abstração de dados: mecanismo que utiliza a proteção de dados e tanto a abstração quanto a proteção de dados aumentam a confiabilidade e facilitam a separação de especificações de procedimentos e representação;
c) coesão dinâmica: indispensável para utilizar um determinado código para outros tipos de dados;

d) hereditariedade: permite a criação de classes que são especializações de outras classes.

Conforme Rocha (2000), autores como Coad, Yourdon, Pressman e tantos outros, abordaram, discutiram e definiram em suas publicações conceitos como:

a) a orientação a objetos é uma tecnologia para a produção de modelos que especifiquem o domínio do problema de um sistema;
b) quando construídos corretamente, sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e provêm a oportunidade de criar e implementar componentes totalmente reutilizáveis;
c) modelos orientados a objetos são implementados convenientemente utilizando uma linguagem de programação orientada a objetos. A engenharia de software orientada a objetos é muito mais que utilizar mecanismos de sua linguagem de programação, é saber utilizar a melhor forma possível todas as técnicas de modelagem orientada a objetos;
d) a orientação a objetos não é só teoria, mas uma tecnologia de eficiência e qualidade comprovadas usada em inúmeros projetos e para construção de diferentes tipos de sistemas.

## 4.1 CONCEITOS BÁSICOS DA ORIENTAÇÃO A OBJETOS

A tecnologia orientada a objetos fundamenta-se nos seguintes conceitos:

a) objeto: é uma ocorrência específica (instância) de uma classe e é similar a uma entidade/tabela no modelo relacional somente até o ponto onde representa uma coleção de dados relacionados com um tema em comum;
b) mensagem: objetos se comunicam através de mensagem, isto é, um sinal enviado de um objeto a outro requisitando um serviço através da execução de uma operação (Furlan,1998);
c) atributos: atributos definem o estado interno de um objeto, suas características, por exemplo, o objeto estante possui as características tamanho, cor, tipo de estrutura;
d) métodos: são códigos para implementação em uma classe, ou operação interna, ou seja, o processo de desenvolvimento. São as funções que operam sobre o objeto (Junior, 1999);

e) classe: é uma coleção de objetos que podem ser descritos com os mesmos atributos e as mesmas operações. Representa uma idéia ou com conceito simples e categoriza objetos que possuem propriedades similares, configurando-se em um modelo para a criação de novas instâncias. Chamamos o objeto de instância de uma determinada classe, o que significa que ele possui o comportamento e as características definidos pela classe para suas instâncias, processo chamado de instanciação;
f) solicitações: para fazer com que um objeto faça alguma coisa, é necessário enviar a ele uma solicitação. Essa solicitação faz com que uma operação seja ativada. A operação executa o método adequado e, opcionalmente, devolve uma resposta (Furlan, 1998);
g) abstração: é a capacidade de olhar apenas uma parte de um todo, ou seja, retirar da realidade apenas as entidades e fenômenos considerados essenciais, excluindo-se todos os aspectos irrelevantes ou secundários;
h) encapsulamento: é o processo de combinas tipos de dados e funções relacionadas em um único bloco de organização e só permitir o acesso a eles através de métodos determinados. Combinação de atributos e operações em um classe (Rocha, 2000);
i) herança: é a capacidade um novo objeto tomar atributos e operações de um objeto existente, permitindo criar classes complexas sem repetir código – a nova classe simplesmente herda seu nível base de características de um ante passado na hierarquia de classe. Também pode reimplementar (ou sobreescrever) operações selecionadas, já que subclasses tiram proveito do comportamento das classes acima na hierarquia passando propriedades de uma classe pai a uma classe filha.
j) polimorfismo: se refere aos vários comportamentos que uma mesma operação pode assumir, assim como à capacidade de uma variável referir-se a objetos diferentes que preenchem certas responsabilidades dependendo da mensagem que lhes é passada. Habilidade para usar a mesma mensagem para invocar comportamentos diferentes do objeto;
k) generalização: atributos e operações comuns compartilhados por classes;
l) especialização: atributos e operações diferentes de uma subclasse, acrescentando ou substituindo características herdadas da classe pai.

# 5 METODOLOGIA OOHDM – OBJECT ORIENTED HIPERMEDIA DESIGN METHOD

O campo da hipermídia se encontra em rápido crescimento e novas áreas de aplicação necessitam da flexibilidade da hipermídia unida com a rica variedade de tipos de dados multimídia. Dessa forma, aplicações que se utilizam de multimídia, como web sites e CD-Roms interativos, apresentam a necessidade de um método ou forma de melhorar o seu planejamento, modelagem e construção. O Método Orientado a Objetos para Design Hipermídia – OOHDM, é uma alternativa para a organização do planejamento de *web sites* e projetos de aplicações multimídia em geral, procurando facilitar as tarefas de criação, manutenção e melhorias.

O processo de construção de aplicações hipermídia não é intrinsicamente diferente daquele utilizado na construção de aplicações convencionais. Segundo Schwabe(2000), existe um crescente consenso acerca do tipo de atividades que devem ser desempenhadas com respeito ao produto de software.

No OOHDM uma aplicação é construída em um processo de quatro passos, para suportar um modelo incremental ou de protótipo. A Tabela 5.1 mostra os quatro passos e as informações relevantes de cada um.

**TABELA 5.1 – METODOLOGIA OOHDM**

| Atividades | Produtos | Formalismo | Mecanismos |
|---|---|---|---|
| **Projeto Conceitual** | Classes, Sub-Sistemas, Relacionamentos, Perspectivas de atributos | Modelagem orientada a objetos | Classificação, Agregação, Generalização e Especialização |
| **Projeto Navegacional** | Nós, *Links*, Estruturas de acesso, Contexto navegacional | *Views*, *State Charts*, classes de Contexto | Classificação, Agregação, Generalização e Especialização |
| **Projeto de Interface Abstrata** | Objetos de Interface abstrata, respostas a eventos externos, Transformações de interface | *Abstract Data View* (ADV), Diagramas de configuração, ADV- charts | Mapeamento entre navegação e Objetos percecptíveis |
| **Implementação** | aplicação | | |

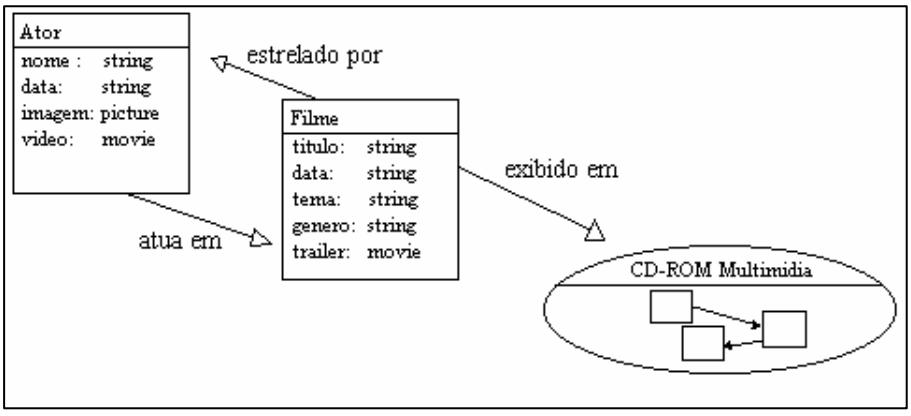
FONTE : Schwabe (2000).

A modelagem conceitual ou de domínio destina-se à compreensão do domínio do problema e à construção de modelos adequados deste domínio, enquanto o projeto lida com abstrações no universo do software e tende à maximização da modularidade e do reuso. O modelo do projeto é independente da implementação no sentido em que, embora possa levar em consideração algumas configurações de implementação, não é condicionado por um ambiente de implementação em particular. Durante a implementação, abstrações de projeto são convertidas em artefatos concretos de implementação, isto é, aqueles existentes em um ambiente de implementação (Pompermaier, 2000).

## 5.1 PROJETO CONCEITUAL

O Projeto Conceitual se constitui basicamente por classes, relações e subsistemas, conforme demonstra a figura 5.1.1. O modelo conceitual é concebido utilizando as técnicas de modelagem já utilizadas para construção de aplicações orientadas à objetos, com a adição de outros elementos, como perpectivas de atributos. As classes conceituais podem ser montadas utilizando as hierarquias de agregação e generalização/especificação; cada classe pode se relacionar com um subsistema (abstrações de um sistema conceitual completo), desde que o mesmo possua um ou mais pontos de entrada (Schwabe,2000). A principal preocupação neste ponto é de observar e representar a semântica do domínio da aplicação, sem muita preocupação com usuários ou tarefas.

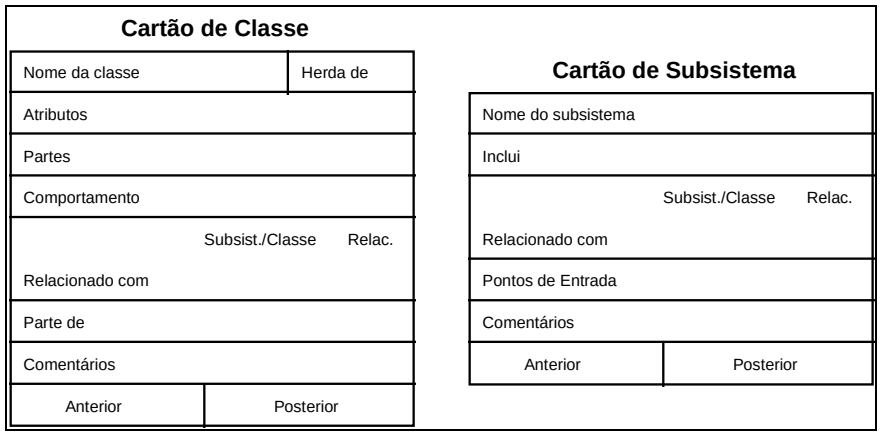
**FIGURA 5.1.1 - MODELO CONCEITUAL COM A REPRESENTAÇÃO DE DUAS CLASSES RELACIONANDO-SE COM UM SUBSISTEMA.**



Para um projeto como este, utilizar as mesmas técnicas utilizadas em orientação à objetos pode trazer várias vantagens e benefícios, como por exemplo os conceitos de herança e polimorfismo e as reações e comunicação entre objetos. Vale lembrar que este tipo de especificação é interessante quando os objetos podem sofrer mudanças constantes, ou haja a necessidade de complexas consultas sobre os mesmos, ou sobre o esquema em si. Os relacionamentos também são uma parte importante desta etapa, servindo como identificação das entidades com maior importância em etapas posteriores (Schwabe, 2000).

São utilizados cartões de identificação, conforme exemplo mostrado na figura 5.1.2, também chamados de cartões de classes e relacionamentos, para facilitar na documentação do sistema. Auxiliam na tomada de decisões, tanto “para frente” como “para trás”, a serem feitas no decorrer do projeto.

**FIGURA 5.1.2 – CARTÕES DE IDENTIFICAÇÃO.**




## 5.2 PROJETO NAVEGACIONAL

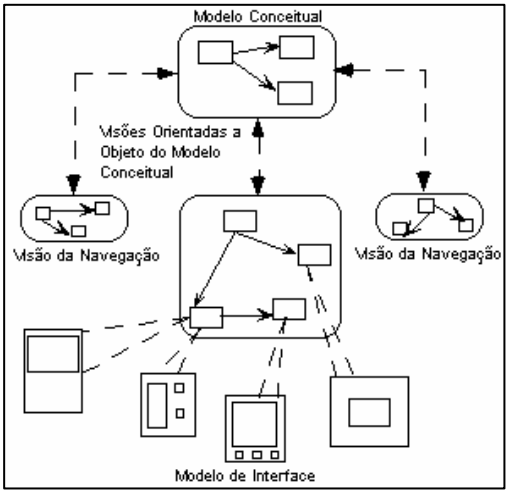
Os aplicativos de hipermídia são projetados para efetuar navegação através de um espaço de informações. Por isto, o projeto da estrutura de navegação de tais aplicativos é a etapa crucial no empreendimento de desenvolvimento.

No método OOHDM, a aplicação é vista como uma visão navegacional sobre o domínio conceitual. Isto demonstra que uma das principais características desta metodologia

que difere das outras é a noção de navegação. Neste ponto, o desenvolvedor deve levar em conta os tipos de usuários e as tarefas que os mesmos irão realizar no uso da aplicação.

A Figura 5.2.1 demonstra a representação do processo, com as visões de navegação sobre o modelo conceitual, e o modelo de interface, interagindo neste conjunto.

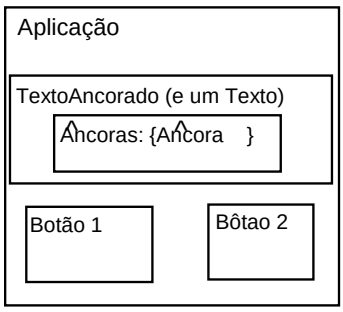
**FIGURA 5.2.1 – REPRESENTAÇÃO DO PROCESSO DE NAVEGAÇÃO**




Diferentes navegacionais pode sem construídos a partir de um projeto conceitual. A estrutura navegacional de uma aplicação hipermídia é definida através de especificações de classes de navegação que refletem as visões sobre o domínio da aplicação. Em OOHDM este projeto é um conjunto de tipos pré-definidos de classes de navegação : **nós, elos, estruturas de acesso**, os quais são organizados em **contextos de navegação** (Schwabe, 2000).

As classes de navegação ou nós, conforme exemplo mostrado na figura 5.2.2, são definidas por um conjunto de atributos provenientes das classes conceituais, como visões orientadas a objetos sobre estas classes. Os nós são uns recipientes de informações básicas.

**FIGURA 5.2.2 - EXEMPLO DE CLASSE DE NAVEGAÇÃO.**



*Links* refletem os relacionamentos que serão utilizados pelo usuário final; são a realização navegacional dos relacionamentos. Classes de *links* são definidas especificando-se atributos, comportamentos do *link*, objetos fonte e alvo e cardinalidade. Atributos do *link* expressam propriedades do próprio *link* e podem ser úteis no momento de definir *links* com uma cardinalidade maior que um. Em alguns casos o *link* pode se comportar como um *nodo*, agindo como um objeto intermediário (entre a fonte e o destino do *link*) durante a navegação.

*Nodos* são as estruturas mais básicas utilizados nas aplicações hipermídia como armazenadores de informação. São visões orientadas a objetos de classes conceituais definidas durante a modelagem conceitual, usando uma linguagem de consulta, permitindo assim que o *nodo* seja definido por uma combinação de atributos de diferentes classes relacionadas no esquema conceitual. Eles contém atributos unitários e *anchors* (âncoras), e podem ser atômicos ou compostos.

As estruturas de acesso são utilizadas para auxiliar o usuário final na procura de uma informação desejada. Menus, Índices e Roteiros Guiados são alguns exemplos de estruturas de acesso. A principal estrutura do esquema navegacional é a noção de Contexto de Navegação. Este contexto constitui-se de um conjunto de nós organizados segundo algum critério e de elos que dão a ordenação a esses nós, formando um caminho de navegação. A Figura 5.2.3 demonstra os possíveis comportamentos do contexto de navegação, segundo Cerqueira (1999).

**FIGURA 5.2.3 – CONTEXTO DE NAVEGAÇÃO**

| Controle \ Restrição de Navegação | | Navegação Livre | Navegação Exclusiva | Navegação Parcial |
|---|---|---|---|---|
| **Automático** | | — | O usuário é guiado automaticamente através dos elos de contexto * | — |
| **Passo a Passo** | **Acesso Direto (Índice)** | O usuário pode acessar quaisquer elementos do contexto de navegação | O usuário só pode acessar elementos do contexto definidos no índice do contexto * | O usuário pode acessar elos "importados" de outros contextos de navegação |
| | **Seqüencial** | O usuário pode percorrer quaisquer elos * | O usuário só pode percorrer elos de contexto | O usuário pode percorrer elos "importados" de outros contextos de navegação |

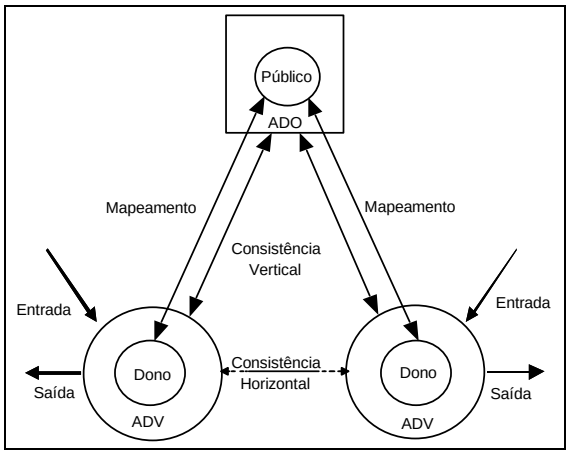
## 5.3 PROJETO DA INTERFACE ABSTRATA

Nesta etapa do processo de modelagem, com a estrutura de navegação já definida, deve-se definir quais objetos irão interagir com o usuário e, principalmente, a maneira como objetos de navegação diferentes vão ser visualizados, quais objetos irão ativar a navegação, a maneira como os objetos de interface multimídia serão sincronizados e quais transformações de interface irão ocorrer. Uma clara separação entre as duas etapas, a interface de navegação e a abstrata, permite a construção de diferentes interfaces para o mesmo modelo de navegação, mantendo assim um alto grau de independência da tecnologia utilizada para criar a interface para o usuário, permitindo também, conformidade com várias necessidades do usuário ou preferências. Na metodologia OOHDM é utilizado o conceito de *Abstract Data View* (ADV), ou Visão de Dados Abstrata, para descrever a interface da aplicação hipermídia.

*Abstract Data View* são objetos que possuem um estado e uma interface, onde a interface pode ser utilizada através de chamadas de funções ou *procedures* regulares, ou eventos de entrada e saída. ADVs são abstratos pois representam apenas a interface e o estado, e não a implementação. Numa aplicação típica usando-se ADV, existe um conjunto de ADOs (*abstract data objects*, ou objetos de dados abstratos) gerenciando estruturas de dados

e controle dentro da aplicação e um conjunto de objetos de interface (instâncias de ADVs) gerenciando aspectos da interface da aplicação como as entradas do usuário e as saídas do sistema para o usuário. Cada ADV deve ser consistente com seu ADO proprietário e todos os ADVs possuídos por um ADO devem ser consistentes entre si. Estes dois tipos de relacionamentos de consistência são chamados, respectivamente, de consistência vertical e horizontal e estão representados na figura 5.3.1 (Schwabe, 2000).

**FIGURA 5.3.1 - DEMONSTRAÇÃO DOS RELACIONAMENTOS DE CONSISTÊNCIA ENTRE ADVS E ADOS.**




No contexto da OOHDM, objetos de navegação como nodos, links ou estruturas de acesso agem como ADOs, e sua ADV associada é usada para especificar a sua aparência para o usuário. Nos próprios ADVs podemos aplicar algumas das técnicas já conhecidas de orientação a objeto, como generalização/composição e herança.

## 5.4 IMPLEMENTAÇÃO

Para executar a implementação, o projetista precisa mapear os modelos de interface abstrata e de navegação em objetos concretos disponíveis no ambiente de implementação escolhido. O modelo gerado depois que todas as etapas anteriores foram concluídas pode ser implementado de uma maneira direta utilizando as várias aplicações para este objetivo já disponíveis no mercado, como por exemplo *Hypercard, Toolbook* e *Director*. O uso de um conjunto de construtores de modelagem (objetos e classes) nesta metodologia permite uma transição suave da modelagem de domínio para o *design* de navegação e interface.

Este passo da implementação não necessita um ambiente orientado à objeto, apesar que o mesmo possa tornar a tarefa mais fácil de ser realizada. No caso do ambiente de execução não ser orientado à objetos, muitas técnicas podem ser usadas para mapear uma especificação orientada à objetos para um ambiente que não seja.

A implementação de uma aplicação hipermídia de modo que esta resulte usável não é tarefa simples. Muitas questões técnicas e não-técnicas devem ser resolvidas. Uma vez que o ambiente de implementação tenha sido escolhido, o projeto deve ser mapeado para artefatos de implementação e todos os componentes hipermídia têm de ser instanciados (Schwabe, 2000).

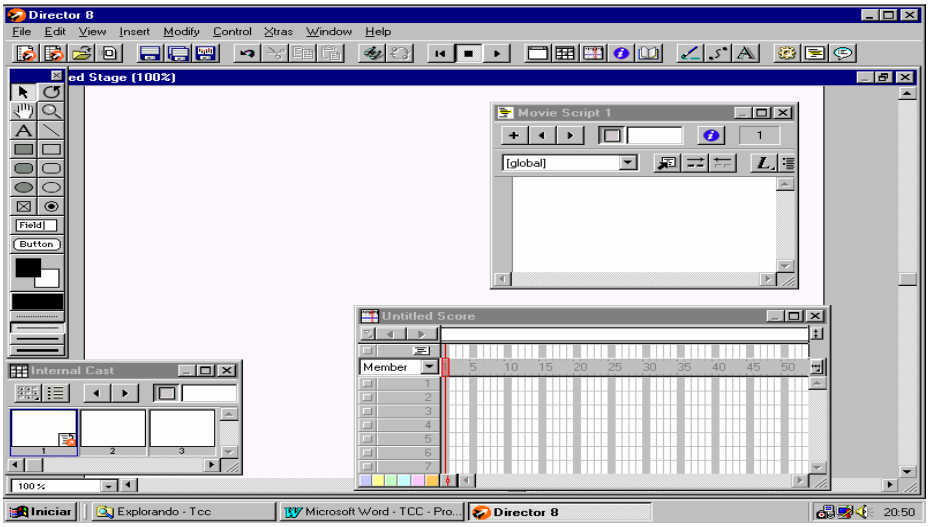
Para manter o conceito de reusabilidade, e aplicá-lo quando necessário, é importante documentar, além de todos os passos das etapas anteriores, as decisões de implementação. Para esta documentação pode-se utilizar o conceito de cartões, que são basicamente fichas que armazenam todas as informações documentacionais da aplicação, de todas as suas etapas, de uma maneira clara e objetiva.

### 5.4.1 AMBIENTE DE IMPLEMENTAÇÃO

O Macromedia Director 8.0 é uma das ferramentas mais utilizadas para o desenvolvimento de softwares hipermídia. Além disso, esta utilização vem aumentando sensivelmente em função da grande evolução do Director nos últimos anos, pois vem incorporando novas e importantes características, principalmente com relação à integração com a Internet. (Bizzotto, 2000).

O desenvolvimento no Director é análogo ao desenvolvimento de um filme de cinema ou de uma peça de teatro. Assim, deve-se definir o tipo de filme (software) a ser desenvolvido; os atores (o elenco) que irão participar; o papel de cada ator; a seqüência de cenas, etc. A figura 5.4.1.1 apresenta alguns dos elementos citados como *Stage* (cenário), *CastMembers* (atores), *Script* (papel de cada ator), *Score* (seqüência de cenas), que serão posteriormente detalhados.

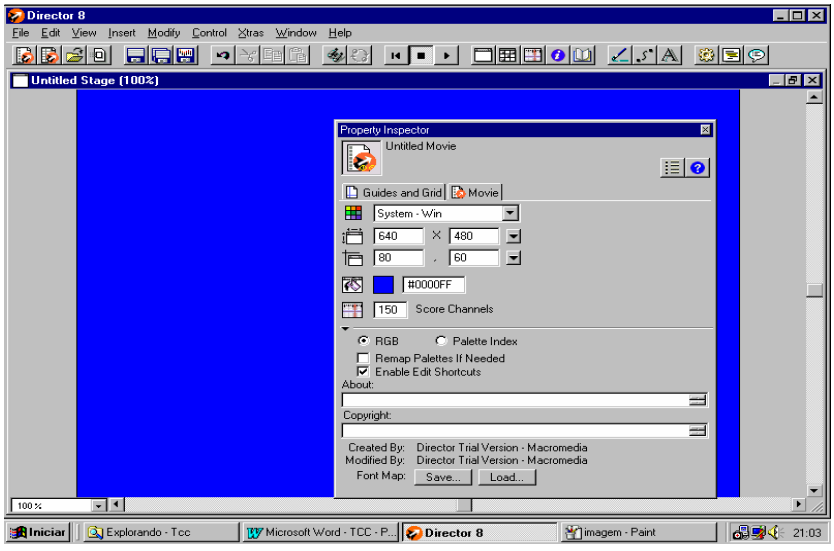
**FIGURA 5.4.1.1 – TELA DE ABERTURA DO DIRECTOR 8.0**




## 5.4.1.1 STAGE

No Director, tudo ocorre no *Stage*, ou seja, no palco (fazendo uma analogia com o teatro). No *Stage*, como mostra a figura 5.4.1.1.1, são colocados todos os itens (cenários, atores etc) que irão compor a cena a ser desenvolvida. No *Stage* podem ser definidas as propriedades do filme atual (*Modify – Movies Properties*), tais como, tamanho do *Stage* de apresentação, a localização, a paleta *Default*, entre outros (Bizzotto, 2000).

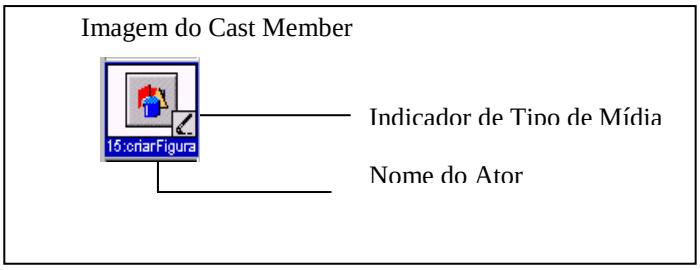
**FIGURA 5.4.1.1.1 – STAGE COM A JANELA PROPRIEDADE DO STAGE**



### 5.4.1.2 JANELA CAST

A janela *Cast*, possui os *castMembers*, que são os atores utilizados por um dado projeto, que atuarão sobre o *Stage*. Desta forma, qualquer imagem, som, vídeo etc, incluído no projeto, irá aparecer na janela *Cast*. A figura 5.4.1.2.1 mostra um membro no *Cast Member*, com suas informações indicadas.

**FIGURA 5.4.1.2.1 – MEMBRO DO CAST.**

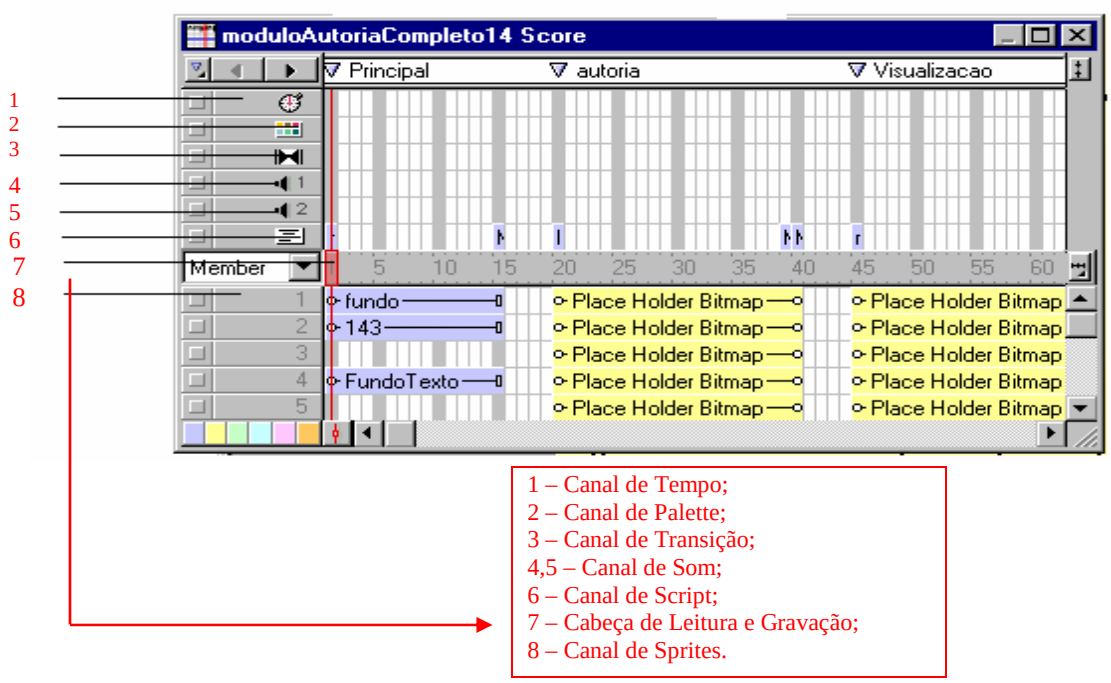



Um mesmo membro do *Cast Member*, pode ser posicionado em locais diferentes do *stage*, sendo chamado de "Instância de *Cast Member*", ou seja, uma cópia do original. Cada *Cast Member* pode ser utilizado de uma vez em uma mesma cena, sendo que cada um deles pode ter comportamento e aparências distintas.

### 5.4.1.3 SCORE

O *Score* pode ser entendido como o roteiro (ou seqüência) do filme onde os *Cast Members* são organizados de acordo com o planejado. O *score* é dividido em linhas (canais) e colunas (*frames*). A intersecção de uma coluna com uma linha, denominados célula. Além dos canais usuais para o gerenciamento dos *Cast Members*, ele possui seis canais especiais para efeitos, conforme mostra a figura 5.4.1.3.1.

**FIGURA 5.4.1.3.1 – CANAIS DO SCORE**

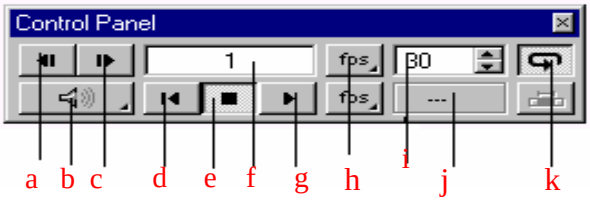



Os itens dispostos na figura 5.4.1.3.1, são responsáveis : (1) pelo gerenciamento do tempo da apresentação e acontecimentos no *Stage*, (2) a possibilidade de utilização de um certo grupo de cores (*palette*) em um dado filme, (3) gerar um efeito de transição de uma cena para outra, (4,5) utilização de sons ou narrações, (6) permitir a programação em Lingo, (7) acompanhar a execução dos *frames*, (8) mostrar todos os *Cast Members* que irão aparecer no *Stage*, respectivamente.

## 5.4.1.4 CONTROL PANEL

A janela *Control Panel* é uma janela especial que funciona como se fosse o controle de um vídeocassete. Além destes controles, esta janela oferece importantes informações sobre o filme. Através desta janela é possível “rodar” um filme (ou parte dele) para avaliar se ele se comporta conforme planejado (Bizzotto, 2000).

**FIGURA 5.4.1.4.1 – JANELA CONTROL PANEL**




Os componentes que compõe o Control Panel, conforme mostra figura 5.4.1.4.1 são:

a) botão *step backward*: retrocede um *frame*;
b) botão *sound*: controla a intensidade de som no filme;
c) botão *step forward*: avança um *frame*;
d) botão *rewind*: retorna para o *frame* 1;
e) botão *stop*: interrompe a exibição do filme;
f) *frame counter*: indica onde se encontra a cabeça de leitura e gravação;
g) botão *play*: inicia a exibição do filme;
h) botão de modo de tempo: permite o andamento do filme em *frames* por segundo;
i) botão modo de tempo real: velocidade real de apresentação do filme;
j) tempo *display*: indica o andamento do filme;
k) *loop playback*: permite que o filme fique em *loop*, executando.

### 5.4.1.5 LINGO

Lingo é o nome da linguagem utilizada pelo Director, para a produção mais efetiva junto ao usuário. O Lingo é um linguagem de "*script*", cuja sintaxe e construção se aproxima da língua inglesa em sua forma usual, ou seja, falada. Os *scripts* podem ser compostos de um único comando, assim como de uma seqüência deles, dispostos em conjuntos, similares a parágrafos. A complexidade varia de acordo com a aplicação.

Como nas demais linguagens de programação, o Lingo segue algumas regras (Henderson, 1997):

a) todo *script* é formado por *handlers*, que em outras linguagens normalmente são definidos como sub-rotinas ou funções. Um *handler* inicia-se com *on* e é encerrado por *end*;
b) variáveis locais: utilizadas somente dentro do *handler* onde foi criada;
c) variáveis globais: utilizadas por diferentes *frames* de um filme, ou de diferentes filmes;
d) ordem de precedência de *handlers*:
   - *scripts* associados a eventos primários tais como:
     - *keydownscript*;
     - *mousedownscript*;
     - *mouseupscript*;
     - *timeoutrscript*;
     - *script* associado a um *sprite* (*sprite script*);
     - *script* associado a um *cast member* (*cast member script);*
     - *script* associado a um *frame* (*frame script*);
     - *script* associado ao filme (*movie script*);
e) orientação a objetos.

Segundo Small (1996), o Lingo permite a programação orientada a objetos, embora não seja considerado uma linguagem orientada a objetos. Através das variáveis globais privadas, cada objeto possui características próprias. A terminologia utilizada na orientação a objeto difere do Lingo, sendo comparados na tabela 5.4.5.1.

**TABELA 5.4.1.5.1 – TERMOS NO LINGO**

| Termo no Lingo | Equivalente em Orientação Objeto |
|---|---|
| *Parent Script* | Classe (*Class*) |
| *Child object* | Instância de classe (*class instance*) |
| *Property variable* | Variável de instância (*instance variable*) |
| *Handler* | Método |
| *Ancestor script* | Super classe |

FONTE : ADAPTADO DE Dalfovo (1997)

#### 5.4.1.6 SCRIPTS

Os *scripts* de Lingo podem variar quanto a sua localização e tipo (Bizzotto, 2000):

a) *behavior script*: são *scripts* associados a *sprites* (*Sprite behaviors*) ou a *frames* (*frame behaviors)*;
b) *cast member script*: associado a um dado *cast member*, de forma que o *script* será executado em qualquer *Sprite* do qual aquele *cast member* faça parte;
c) *movie script*: são *scripts* que "valem" para todo o filme. Assim, eles são mais gerais, não se limitando a um dado *sprite* ou conjunto de *sprites* ou *frames*.

#### 5.4.1.7 EVENTOS

No momento em que o usuário vai utilizar o software, ocorrem os seguintes eventos no instante da inicialização (Bizzotto,2000):

a) *Prepare movie*: este é o primeiro evento que ocorre quando um filme é executado, ou seja, o filme está sendo preparado para ser rodado;
b) *BeginSprite*: indica que a cabeça de leitura e gravação entrou em um segmento de *sprite* (*sprite span*), ou seja, indica que um *sprite* foi iniciado;
c) *PrepareFrame*: ocorre antes do Director apresentar um *sprite* no *stage*. Assim, neste evento pode-se incluir ações que alterem as propriedade do *sprite*, uma vez que ele ainda não foi "desenhado";
d) *StartMovie*: este evento ocorre no primeiro *frame* do filme.

A partir do momento que o evento *StartMovie* ocorreu, o Director passa a executar os *frames* do filme. Enquanto a cabeça de leitura e gravação passa por todos os *frames* do filme, ocorrem os seguintes eventos:

a) *beginSprite*: indica que a cabeça de leitura e gravação entrou em um segmento de *sprite* (*sprite span*). Este evento ocorre sempre que existe um novo segmento de *sprite*;
b) prepareFrame: o frame está sendo preparado para ser "desenhado"no stage;
c) *enterFrame*: ocorre depois que o *sprite* é "desenhado" no *stage* e antes que a cabeça de leitura e gravação deixe o *frame* atual;
d) *exitFrame*: ocorre quando a cabeça de leitura e gravação deixa um dado *frame*;

e) *endSprite*: ocorre quando a cabeça de leitura e gravação deixa a última célula de um segmento de *sprite*.

Estes eventos são independentes da ação do usuário, ou seja, não é necessário a intervenção do usuário para que estes eventos ocorram. Mas, existem eventos que dependem da ação direta do usuário. Estes eventos são:

a) *mouseEnter*: ocorre quando o cursor entra no espaço de um *sprite*;
b) *mouseWithin*: ocorre enquanto o ponteiro do mouse estiver sobre um dado *sprite*;
c) *mouseLeave*: ocorre quando o mouse deixa o espaço de um *sprite*;
d) *mouseDown*: ocorre quando o usuário pressiona o botão do mouse;
e) *mouseUp*: ocorre quando o usuário solta o botão do mouse;
f) *mouseUpOutside*: ocorre quando o usuário pressiona o botão do mouse sobre um *sprite*, mas solta o botão fora deste *sprite*;
g) *rigthMouseDown*: ocorre quando o botão direito do mouse é pressionado;
h) *rigthMouseUp*: ocorre quando o botão direito do mouse é liberado;
i) *keyDown*: ocorre quando o usuário pressiona uma tecla;
j) *keyUp*: quando o usuário solta uma tecla.

#### 5.4.1.8 LISTAS

As listas em Director, ou *arrays* em outras linguagens, são “variáveis” que podem conter diversos elementos. A alocação dos elementos na memória, é feita segundo a necessidade desta, não denso portando, necessárias as informações quanto ao tamanho dos elementos da lista. O primeiro elemento da lista no Director é contado como “1”, diferentemente das demais linguagens que consideram o primeiro elemento de lista“0” (Small, 1999). A tabela 5.4.1.8.1 mostra sintaxes para criação de listas.

**TABELA 5.4.1.8.1 – SINTAXE PARA A CRIAÇÃO DE LISTAS**

| Sintaxe | Criando uma lista |
|---|---|
| *Set the list to* [ ] | Para criar uma lista linear vazia |
| *Set the list to* [:] | Para criar uma propriedade de lista vazia, pode-se usar um operador de lista para especificar valores na lista. |
| Especifique a lista de elementos com os parâmetros da função *list*().<br>*List*(*value1, value2, value3...)* | Para criar uma lista linear usando a função list()<br>Exemplo:<br>*Set designers = list(“Gee”,”Kayne”, “Ohashi*”) |

FONTE : Small (1999)

## 5.5 TESTE E MANUTENÇÃO

Teste e manutenção podem ser integradas com sucesso ao processo de desenvolvimento através do reconhecimento de que o ciclo de vida de um produto de software é um contínuo em que o software evolui na medida em que novas exigências aparecem.

# 6 DESENVOLVIMENTO DO PROTÓTIPO

No desenvolvimento do protótipo foi utilizada a metodologia OOHDM, apresentada no Capítulo 5. O OOHDM utiliza mecanismos de abstração e composição numa arquitetura orientada a objetos, que permite uma descrição concisa das informações complexas e uma especificação de padrões de navegação e interface (Schwabe, 2000).

Conforme Rumbaugh (1994), os sistemas orientados a objetos podem representar melhor o mundo real, uma vez que a percepção e o raciocínio do ser humano estão relacionados diretamente com o conceito de objetos. Este fato permite uma modelagem mais perfeita e natural. Além desta característica, a OO oferece muitos outros benefícios:

a) a mesma notação é usada desde a análise até o projeto e a implementação;
b) esta abordagem incentiva os desenvolvedores a trabalharem e pensarem em termos do domínio da aplicação durante a maior parte do ciclo de vida;
c) ocorre uma redução na quantidade de erros com conseqüente diminuição do tempo despendido nas etapas de codificação e teste;
d) no desenvolvimento baseado em objetos há uma redução no tempo de manutenção, pois as revisões são mais fáceis e mais rápidas, já que o problema é melhor localizado;
e) favorece a reutilização;
f) facilidade de extensão, visto que objetos têm sua interface bem definida;
g) Flexibilidade, produtividade e qualidade; aplicação.

As seções seguintes abordarão cada um dos passos para a realização do desenvolvimento do protótipo de aplicação hipermídia utilizando OOHDM.

## 6.1 PROJETO CONCEITUAL

Em uma modelagem orientada a objetos procura-se representar os aspectos estruturais e comportamentais que serão mapeados para o ambiente de implementação como estruturas de classes com seus métodos.

O presente protótipo visa atender as seguintes necessidades:

a) facilitar ao professor a elaboração e correção de provas e exercícios, adequando-os ao conteúdo exposto durante as aulas, conforme a metodologia de ensino adotada

por ele, uma vez que a quantidade de alunos dificulta o processo de avaliação de desempenho individual, pelo tempo dispendioso que é exigido para sua execução;

b) passar a associar nas escolas, o uso do computador a complementos disciplinares ou projetos educacionais;
c) motivar o professor à capacitação, percebendo como deve efetuar a integração da tecnologia com a sua proposta de ensino, ficando aberto à mudanças, principalmente quanto a sua postura de negar-se a precisar aprender;
d) motivar o aluno através do uso da informática através do jogo de quebra-cabeça a praticar o conteúdo trabalhado.

O público alvo do protótipo a ser desenvolvido são professores de qualquer área do conhecimento. O perfil geral destes professores é o que segue:

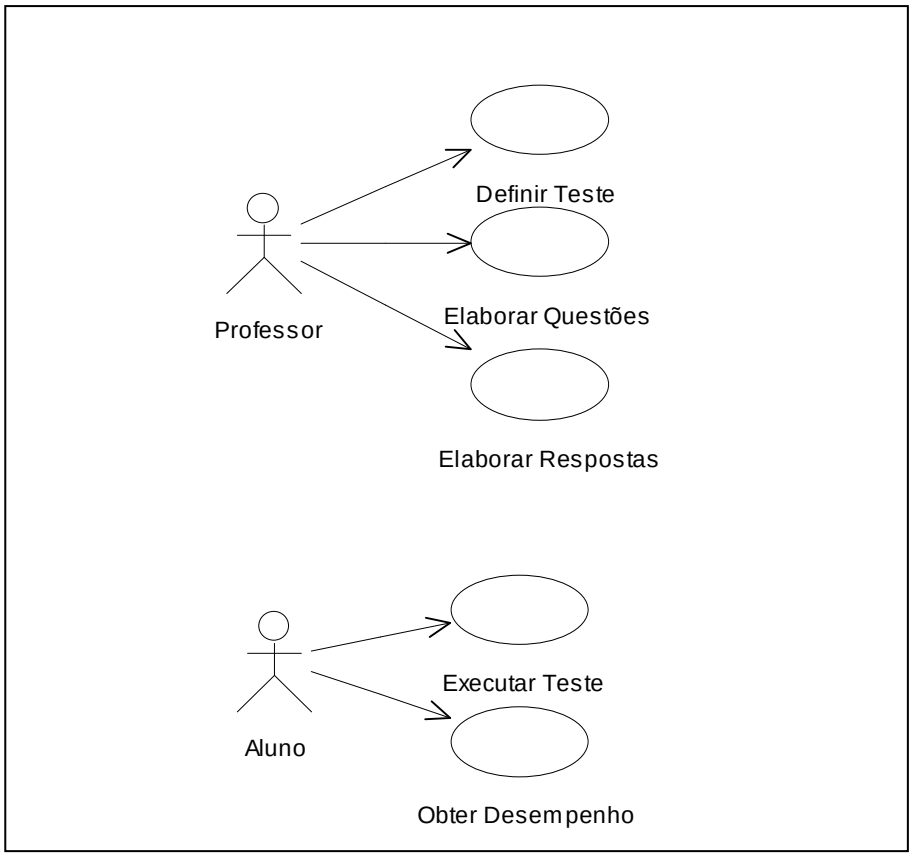
a) conhecimento básico da informática;
b) conhecimento pedagógico;
c) saber identificar o elo entre estas duas áreas;
d) conhecer formas de gerenciar uma sala de aula com esses novos recursos tecnológicos, tanto recursos físicos como o novo comportamento dos alunos, que passam a ter uma atitude mais ativa neste processo;
e) é necessário que o professor faça uma revisão das teorias de aprendizagem, didática, construção do conhecimento, interdisciplinaridade e forma de abordagem da aprendizagem significativa. Estar aberto para as mudanças, conscientizando-se que precisa aprender a aprender;
f) é necessário que os professores criem um canal de comunicação entre eles, para troca de informações e experiências, amenizando seus receios quanto ao uso da informática;

O domínio da aplicação é criado utilizando os princípios de modelagem orientada a objetos apresentada em Furlan (1998), a UML.

### 6.1.1 DIAGRAMA DE CASOS DE USO (USES CASE)

Nesta fase serão identificados os atores e casos de uso que farão parte do protótipo e especificarão a sua funcionalidade, como mostra a figura 6.1.1.1.

**FIGURA 6.1.1.1 - DIAGRAMA DE CASO DE USO**



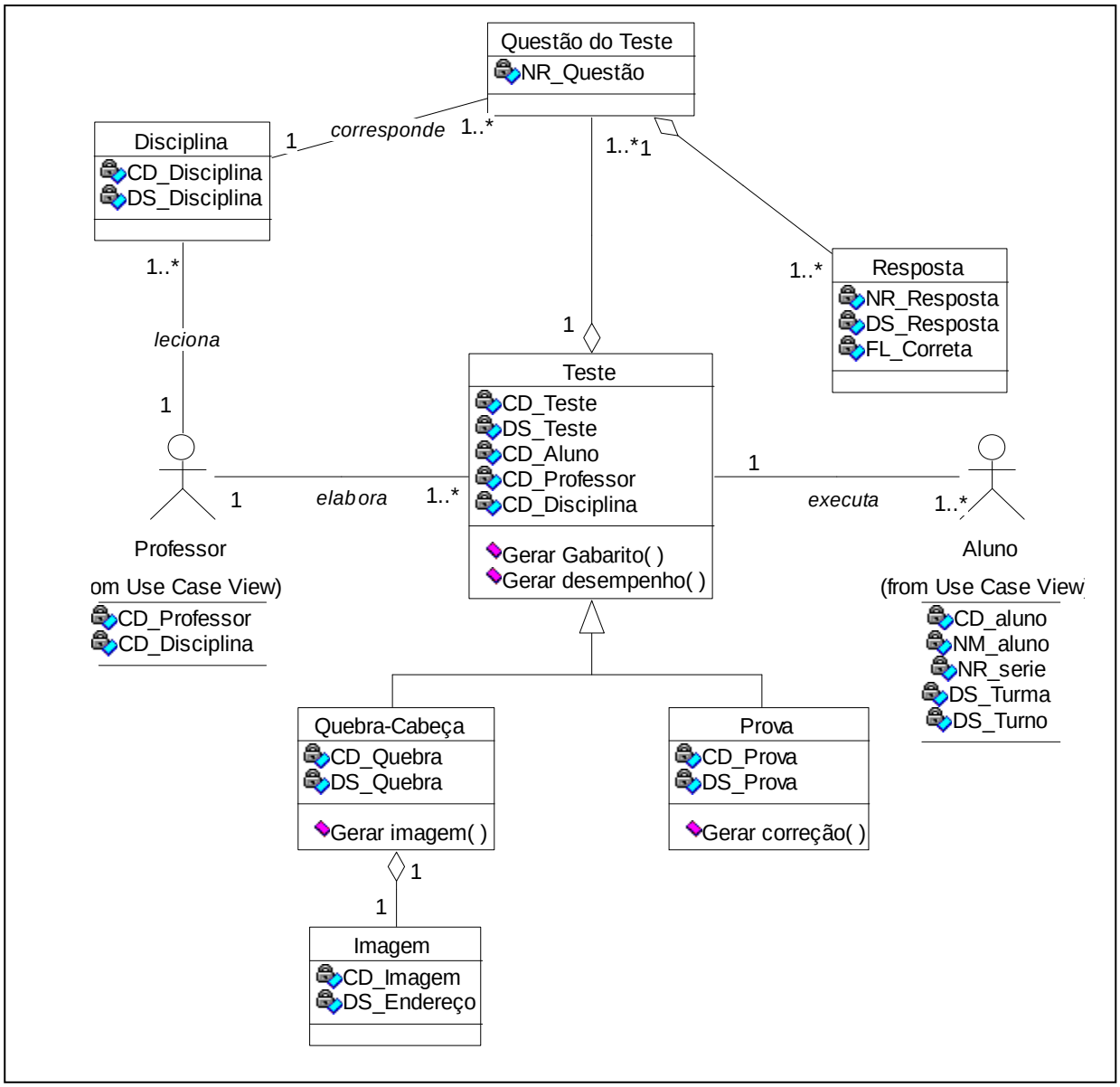

Os casos de uso são:

a) definir tipo de teste : o professor deverá definir qual o teste que o aluno deverá executar para fixar o conteúdo. quebra-cabeças ou prova;
b) elaborar questões : o professor elabora as questões que farão parte do teste definido; Para o exercício de quebra-cabeça, o professor deverá definir imagens que comporão a solução do mesmo;
c) elaborar respostas : o professor elabora o gabarito do teste, conforme definido, sendo várias respostas para o teste de múltipla escolha, imagem solução para o quebra-cabeça;
d) executar Teste : o aluno executa o teste proposto pelo professor, que poderá ser exercício ou prova, para fixar o conteúdo exposto;
e) verificar desempenho : o aluno, após executar o teste, poderá verificar o seu desempenho;

## 6.1.2 DIAGRAMA DE CLASSES

O Projeto Conceitual se constitui basicamente por classes, relações e subsistemas. O modelo conceitual é concebido utilizando as técnicas de modelagem já utilizadas para construção de aplicações orientadas à objetos, com a adição de outros elementos, como perspectivas de atributos. As classes conceituais podem ser montadas utilizando as hierarquias de agregação e generalização/especificação; cada classe pode se relacionar com um subsistema (abstrações de um sistema conceitual completo), desde que o mesmo possua um ou mais pontos de entrada (Schwabe, 2000).

**FIGURA 6.1.2.1 - DIAGRAMA DE CLASSES**



Conforme mostra a figura 6.1.2.1, diagrama desenvolvido com o auxílio da ferramenta *Case Rational Rose*, o professor elabora um ou vários testes, de acordo com o conteúdo pertencente a disciplina que leciona. Este teste poderá ser uma avaliação (prova) do conteúdo exposto, ou simplesmente um jogo de quebra-cabeça. Se o professor elaborar uma prova, a este teste estão agregadas as questões, quantas eles quiser inserir, e as várias alternativas de respostas. Inserindo também as respostas certas para cada questão, gerando um gabarito para este teste. Se for definir um jogo de quebra-cabeça, o professor escolhe a imagem pertinente ao conteúdo exposto.

O aluno executa o teste proposto pelo professor. Montar um quebra-cabeça, que a partir da imagem inicial, são geradas várias peças de tamanhos iguais e posicionadas na tela aleatoriamente,  ou executar uma prova. Ao final da execução do teste, é gerada a correção e obtido o desempenho do aluno.

## 6.2 PROJETO NAVEGACIONAL

As aplicações hipermídias são projetadas para realizar navegação através de um espaço de informações. A navegação é uma característica das aplicações hipermídia em que o usuário pode percorrer informações que se apresentam na tela do computador. Esta é uma das principais característica destes tipos de sistemas.

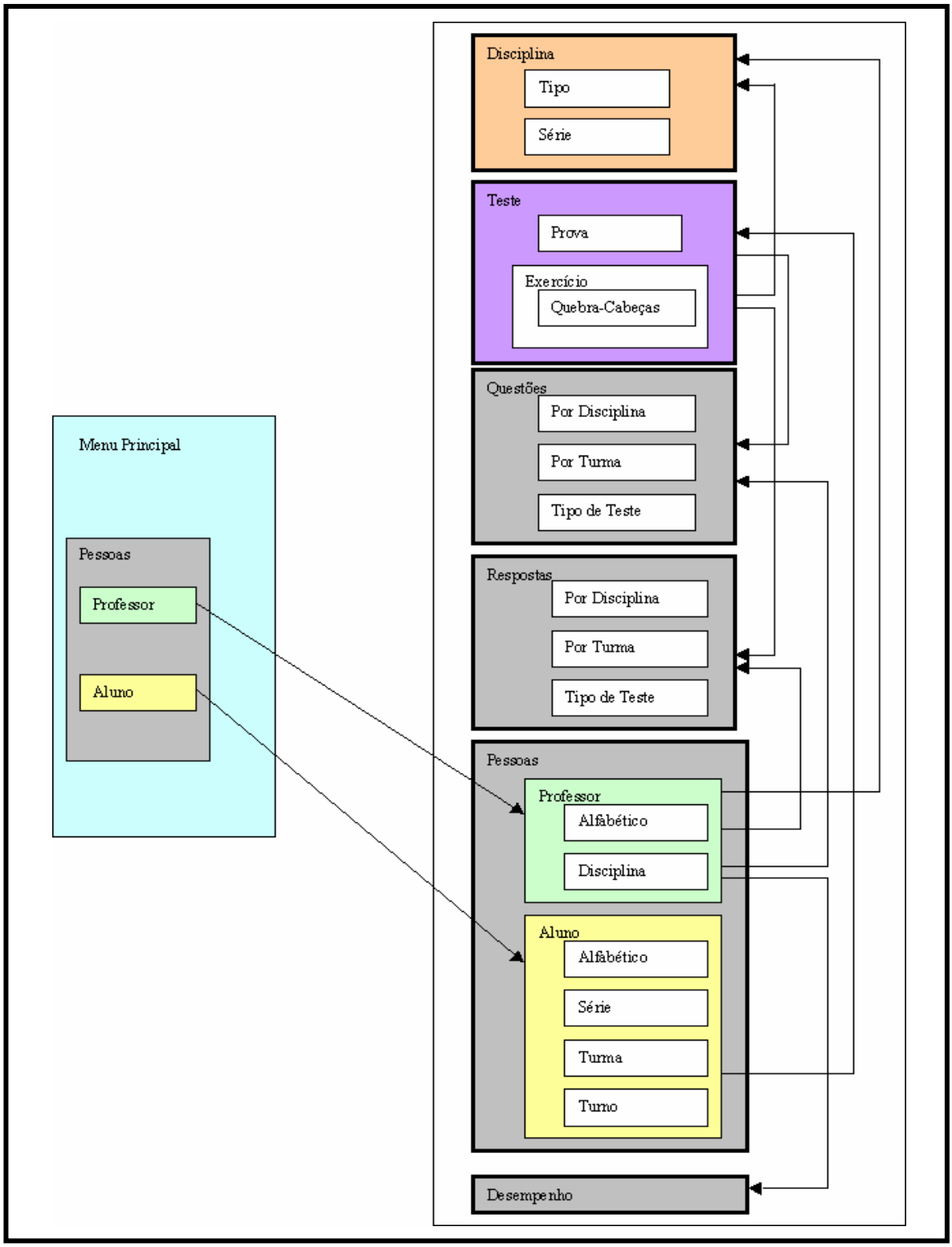
O ambiente de implantação do protótipo é em ambiente educacional, abrangendo todas as áreas do conhecimento. O conteúdo a ser utilizado no protótipo é :

a) **módulo do professor :** onde somente o professor terá acesso, para a correção e avaliação dos exercícios ou provas executadas pelos alunos;
b) **módulo do aluno :** onde o aluno terá acesso para executar a tarefa estabelecida pelo professor, prova ou exercício;
c) **módulo das questões :** onde somente o professor terá acesso para digitar as questões com testes de múltipla escolha, perguntas subjetivas, referentes a prova que irá propor ao aluno ou escolher imagem para o jogo de quebra-cabeça;
d) **módulo das respostas :** onde o professor digita o gabarito das questões para que o protótipo possa corrigir a prova executada e mostrar automaticamente o resultado do desempenho do aluno;

e) **módulo desempenho** : onde o protótipo compara as questões com o gabarito e mostra ao professor o desempenho do aluno na tarefa que executou.

A Figura 6.2.1 demonstra o esquema navegacional do protótipo.

**FIGURA 6.2.1 – PROJETO NAVEGACIONAL**

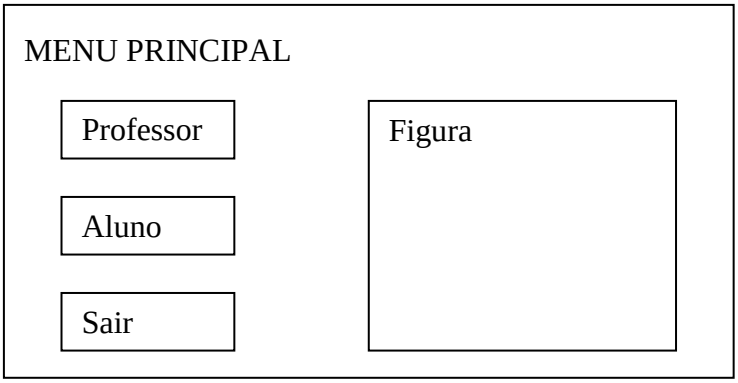


## 6.3 PROJETO DE INTERFACE ABSTRATA

Os Diagramas de Configuração são úteis na expressão de padrões de comunicação entre os objetos em termos de serviços oferecidos e requeridos (Schwabe, 2000). No contexto de OOHDM é importante o modo como o usuário irá interagir com a aplicação e, especialmente, quais objetos de interface causarão a navegação. A seguir serão demonstrados os diagramas de configuração das Classes do protótipo.

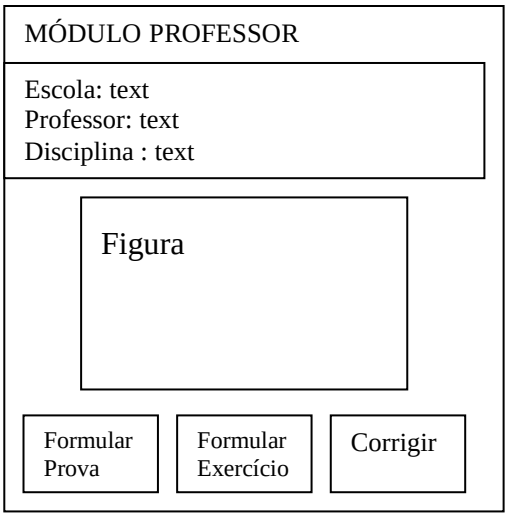
Na primeira etapa, conforme mostra a figura 6.3.1, o professor ou o aluno, selecionam um módulo. No caso de professor, para elaborar provas e exercícios e fazer correção, e no caso do aluno para efetuar a tarefa proposta pelo professor (executar prova ou exercício).

**FIGURA 6.3.1 – DIAGRAMA DE CONFIGURAÇÃO DO MENU PRINCIPAL**



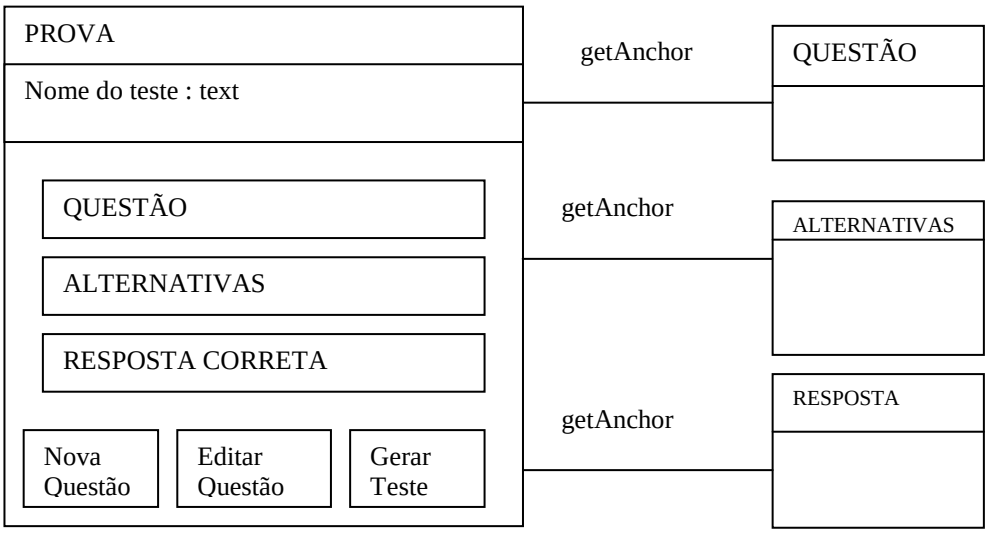

Selecionado o módulo professor, na tela seguinte, conforme mostra a figura 6.3.2, o usuário digita o nome da escola, seu nome e a disciplina referente a prova ou exercício que irá elaborar. O módulo corrigir é uma opção executada após terem sido feitos os testes pelos alunos e o professor obtém através do mesmo, a avaliação de cada aluno que efetuou o teste proposto.

**FIGURA 6.3.2 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO PROFESSOR**



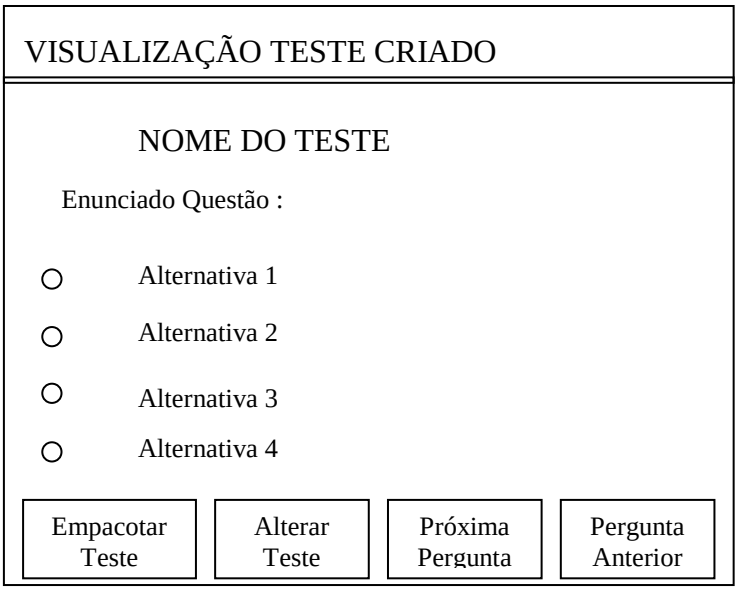

Na próxima etapa, conforme mostra a figura 6.3.3, o protótipo carrega o nome do teste, que é digitado quando o usuário escolher a opção de executar prova. O professor que irá elaborar a prova vai incluindo na mesma, as questões, alternativas e respostas corretas, na quantidade que achar necessário. O protótipo verifica e armazena estas questões, alternativas e respostas. Terminada a elaboração da prova, o professor gera a prova, onde o protótipo a armazena, gerando-a no formato que aparecerá na tela do aluno que irá executá-la.

**FIGURA 6.3.3 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO PROVA**



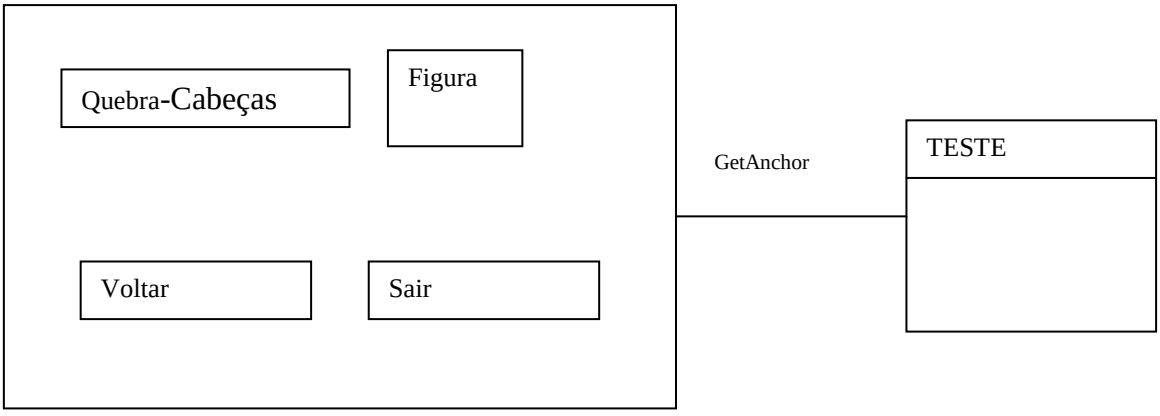
Após solicitar a opção de gerar teste, o professor poderá visualizar a prova que elaborou, conforme mostra a figura 6.3.4. Podendo alterar a mesma, e navegar entre as questões para visualização. Se estiver conforme, o professor empacota o teste, ou seja, grava em uma pasta o arquivo contendo a prova, ao qual o aluno terá acesso para executá-lo.

**FIGURA 6.3.4 – VISUALIZAÇÃO DO TESTE CRIADO**



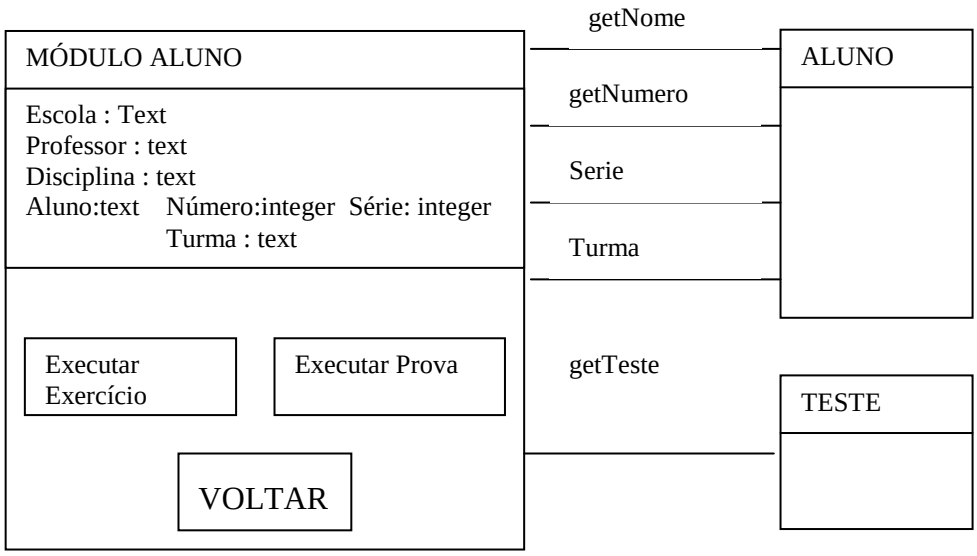

Se o professor selecionar a opção de elaborar exercício, o protótipo apresenta a tela do exercício de quebra-cabeças, onde o professor seleciona uma imagem em qualquer pasta e encaminha para o módulo aluno. Esta etapa é subseqüente a opção formular exercício. A figura 6.3.5 mostra a tela de seleção do exercício.

**FIGURA 6.3.5 – DIAGRAMA DE CONFIGURAÇÃO DO SUB-MENU DO MÓDULO FORMULAR EXERCÍCIO.**



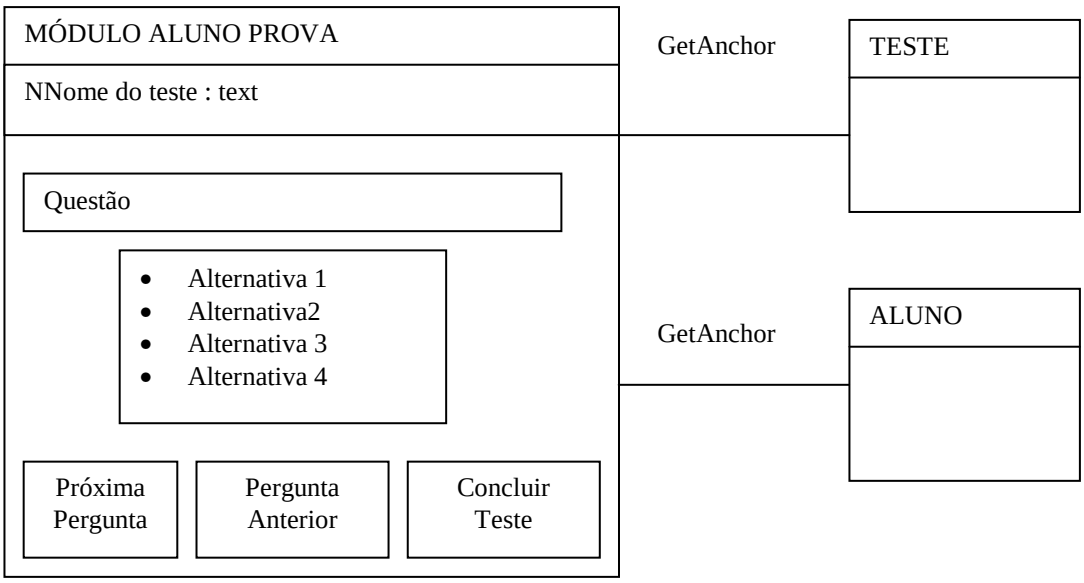
Esta etapa é subseqüente a tela inicial do protótipo, após selecionar o módulo do aluno. O usuário entra com os dados correspondentes para que o protótipo possa verificar e gravar. Quando o aluno entra neste módulo terá duas opções, dependendo da tarefa que for designada a ele. Qualquer uma das opções está agregada a um teste elaborado anteriormente pelo professor. A figura 6.3.6 mostra a tela de seleção de tarefa do módulo aluno.

**FIGURA 6.3.6 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO ALUNO.**



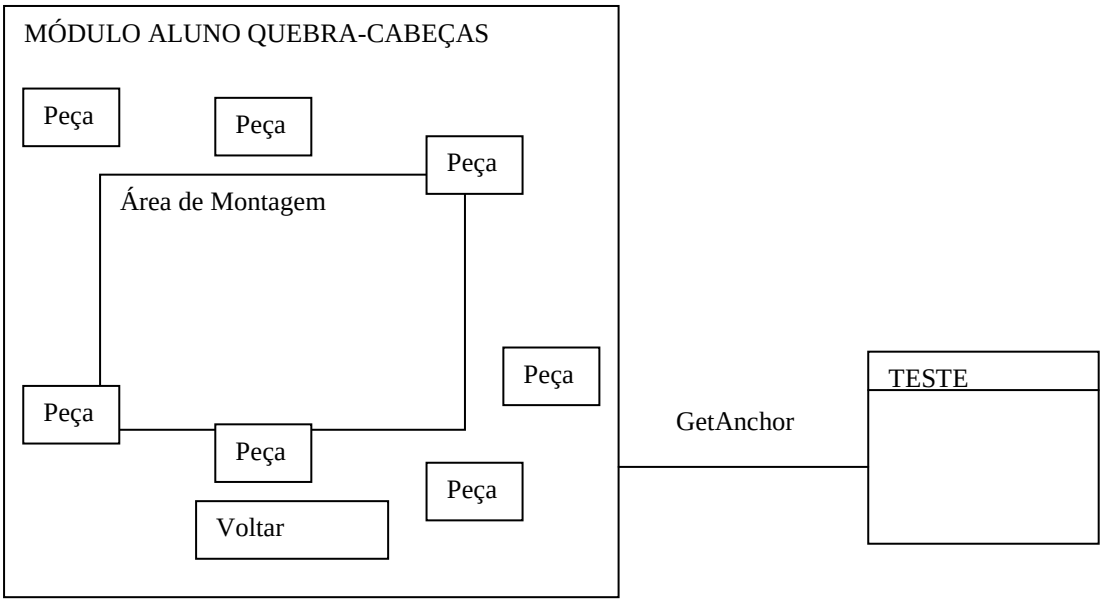

Se o aluno optar por executar uma prova, aparecerá na tela o teste previamente elaborado pelo professor, conforme mostra a figura 6.3.7. Quando o aluno inicia a execução do teste, cada tela apresenta a questão e alternativas de resposta, onde o aluno clica na resposta que achar correta, podendo retroceder e passar adiante, até concluir o teste. O protótipo grava a questão com a resposta que o aluno clicou, para que o desempenho seja avaliado, sendo comparado com o gabarito inserido pelo professor no módulo correção.

**FIGURA 6.3.7 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO ALUNO EXECUTAR PROVA.**



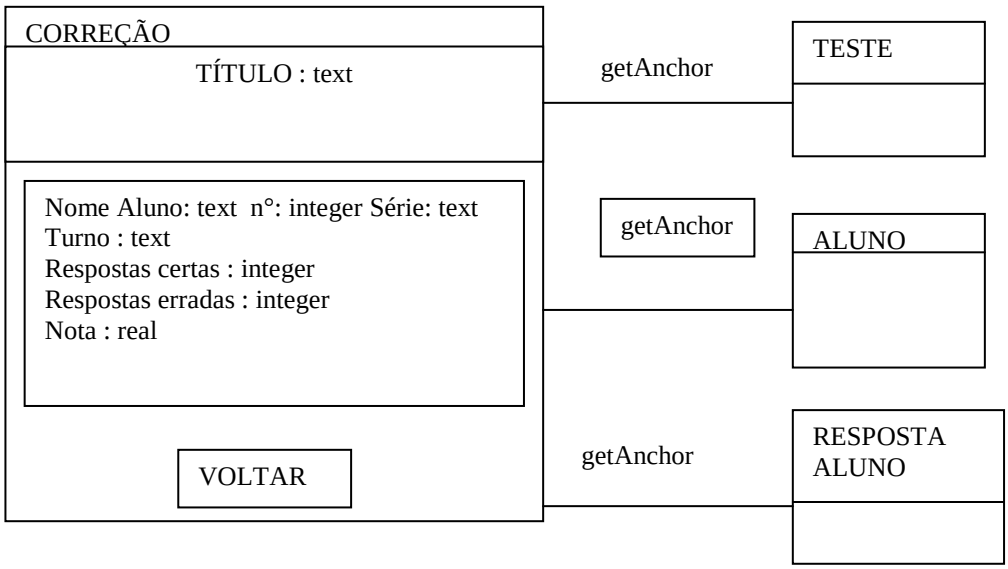

Nesta etapa, a imagem aparecerá recortada em peças de tamanhos iguais e posicionadas na tela aleatoriamente. O aluno vai clicando e arrastando para a área de montagem cada uma das peças, até concluir a imagem designada. Cada peça colocada no lugar errado, o protótipo não a encaixa corretamente. A figura 6.3.8 mostra a tela do exercício de quebra-cabeças.

**FIGURA 6.3.8 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO ALUNO EXECUTAR EXERCÍCIO QUEBRA-CABEÇAS.**



Neste módulo, de acesso exclusivo do professor, o usuário acessa para fazer a correção das provas executadas pelos alunos e obtém as notas correspondentes. O professor indica a pasta onde estão os testes executados pelos alunos, no momento em que clicar na opção de Corrigir Prova.. O protótipo efetua a correção, comparando as respostas dos alunos com o gabarito inserido pelo professor e calcula as notas, mostrando na tela o nome do aluno, número, série, turno, com as respostas certas, respostas erradas e a sua respectiva nota. A figura 6.3.9 mostra a tela de resultado da correção de prova.

**FIGURA 6.3.9 – DIAGRAMA DE CONFIGURAÇÃO DO MÓDULO CORREÇÃO**



## 6.4 IMPLEMENTAÇÃO

Para obter a implementação da aplicação, deve-se mapear os modelos navegacionais e interface abstrata em objetos concretos, em um ambiente de implementação.

Nesta fase é realizado o mapeamento dos objetos perceptíveis e suas transformações, definidos no projeto da interface abstrata para os projetos disponíveis no ambiente de software de autoria hipermídia, o Director 8.0, selecionado para a implementação do protótipo.

Houve a criação do código no ambiente escolhido e em conjunto a esta atividade, foram criadas as imagens, formatação do texto, botões, *backgrounds*, enfim, os aspectos gráficos do protótipo. Conforme a interface abstrata, foram definidos os aspectos gerais de cada tela. A seguir, estão dispostos alguns códigos implementados com suas respectivas telas. A figura 6.4.1 mostra a tela de entrada do protótipo. Esta tela é uma tela de transição, sendo precedida pelo menu principal.

**FIGURA 6.4.1 - TELA INICIAL DO PROTÓTIPO**

Após a tela de transição, o protótipo apresenta o menu inicial, conforme mostra a figura 6.4.2.

**FIGURA 6.4.2 – MENU PRINCIPAL DO PROTÓTIPO**


Esta tela é um menu que permite o acesso do usuário a um dos dois módulos que o protótipo apresenta. Sendo o usuário, professor, optará pelo módulo professor para elaborar os exercícios e provas ou fazer correções. Sendo o usuário aluno, optará pelo módulo aluno para executar a tarefa pré-definida pelo professor. Esta tela é uma composição de imagens *.GIF* que foram importadas, mais os botões criados através de ferramentas do ambiente e *behaviors* de navegação. O quadro 6.4.1 , apresenta os *behaviors* de navegação dos botões módulo professor e módulo aluno, e um *behavior* principal utilizado para navegar em algumas das tela do protótipo. Os *behaviors* são *scripts* responsáveis por cada objeto disposto na tela. A vantagem da sua utilização e que podem ser associados a mais de um objeto.

**QUADRO 6.4.1 – BEHAVIORS DE NAVEGAÇÃO**

| **BOTÃO MÓDULO PROFESSOR** | **BOTÃO MÓDULO ALUNO** |
|---|---|
| on mouseUp me<br>  go to frame "Menu"<br>end | on mouseUp me<br>  go to movie "provaModuloAluno"<br>end |

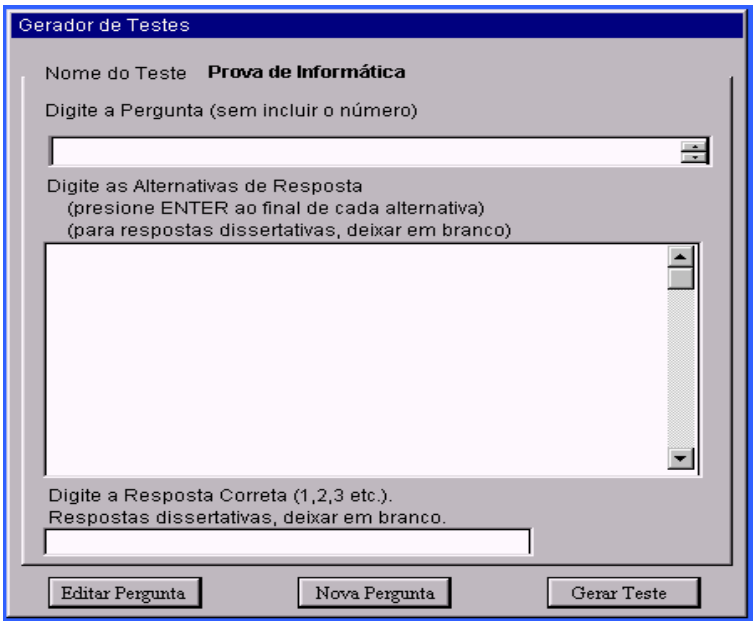
```
property pMarcador
on getPropertyDescriptionList me
  propDescList = [:]
  propDescList[#pMarcador] = [#comment:"Marcador",\
                #format:#marker,#default:""]
  RETURN propDescList
end getPropertyDescriptionList me
on mouseUp me
  go frame pMarcador
end mouseUp me
```

Ao selecionar o botão módulo Professor, tela seguinte mostrada na figura 6.4.3, o professor terá três opções de escolha, dependendo da tarefa que queira executar, navegando para a tela subseqüente.

**FIGURA 6.4.3 – TELA DE SELEÇÃO DE TAREFA DO PROFESSOR**

Se for selecionada a opção de formular prova, aparecerá uma tela para digitar o nome da prova e navegar para a tela de gerador de teste, onde o usuário passa a elaborar o teste. Sempre incluindo as questões, alternativas de respostas e a respectiva resposta correta. Tendo as opções de editar cada questão ou incluir mais questões até a quantidade que quiser, conforme mostra a figura 6.4.4. Ao selecionar a opção de nova pergunta, o protótipo inclui as perguntas, alternativas e respostas digitadas em uma lista e limpa a tela atual. Se escolher a opção de gerar teste, o protótipo gera o teste, montando para visualização do professor e empacota para o aluno executar.

**FIGURA 6.4.4 – TELA GERADOR DE TESTES**




O quadro 6.4.2, mostra parte do código da opção de formular prova, quando o professor está inserindo as questões.

**QUADRO 6.4.2 – PARTE DO CÓDIGO DO BEHAVIOR DE FORMULAR PROVA**

```
on mCriarMembers(me)

  -- Identificando o número da pergunta
  tLista = pListaTeste[1]
  tNumeroPergunta = (tLista.count() + 1)

  -- Gerando o nome do cast member com a pergunta
  tNomeMemberPergunta = pNomeTeste & "Pergunta" & tNumeroPergunta

  me.mVerificarSeExiste(tNomeMemberPergunta)

  -- Criando um novo cast member para a pergunta
  pNovaPergunta = new(#text,castLib "Midias")

  -- Definindo o nome
  pNovaPergunta.name = tNomeMemberPergunta

  -- Definindo o texto da pergunta
  pNovaPergunta.text = member(pMemberPergunta,"geradorPerguntas").text

  -- Identificando se é uma resposta objetiva ou dissertativa
  if pMemberAlternativas.text <> "" then
    tNumeroDeAlternativas = the number of lines in pMemberAlternativas.text
    pListaAlternativas = []
    repeat with i = 1 to tNumeroDeAlternativas
        if pMemberAlternativas.text.line[i] <> "" then
         tNomeAlternativa = tNomeMemberPergunta & "Alternativa" & i
         me.mVerificarSeExiste(tNomeAlternativa)
         tMemberAlternativa = new(#button,castLib "Midias")
         tMemberAlternativa.buttonType = #radioButton
         tMemberAlternativa.name = tNomeAlternativa
         tMemberAlternativa.text = pMemberAlternativas.line[i]
         pListaAlternativas.append(tMemberAlternativa)
       end if
    end repeat
  else
    pListaAlternativas = [#dissertativa]
  end if
  pNovaResposta = pMemberRespostaCerta.text
end mCriarMembers
```

Cada questão, alternativa e resposta certa de um teste, são gravadas em *cast members*, sendo nomeados corretamente e feitos testes para evitar duplicidade. Cada item posicionado no *stage*, é considerado ator ou *cast member* pelo Director.

As questões, alternativas de respostas e respostas certas inseridas pelo usuário, são também gravadas em uma lista, conforme mostra o quadro 6.4.3, e esta lista é gravada em um *script* denominado “Quadro Negro”, para quando o aluno pedir para executar a prova, o protótipo possa recuperar o teste deste *script* e gerar as telas para posiciona-las na tela do aluno, no formato adequado. Deste mesmo *script* é recuperado o teste para posicionar na tela

de visualização do teste gerado no módulo professor, para que o professor possa visualizar o teste e fazer as devidas alterações quanto estiver formulando o mesmo e empacotar para mandar para o aluno.

**FIGURA 6.4.3 – PARTE DO CÓDIGO DO BEHAVIOR PARA GERAR TESTE**

```
on beginSprite(me)
  pMeuSprite = me.spritenum
  pMeuMember = sprite(pMeuSprite).member
end beginSprite

on mouseUp(me)
  -- Obtendo a lista com as perguntas, as alternativas de resposta e a
resposta certa
  tLista = mObterListaTeste(script "quadroNegro")

  tListateste = tLista[1]
  go to frame "visualizar"

  -- Se a lista com perguntas e respostas estiver vazia...
  if tLista.count() = 0 or voidP(tLista) then

    -- ...então não foi gerado um novo teste
    go to frame "Menu"
    abort

  else

    me.mGerarTelasTeste(tListaTeste)

  end if

end mouseUp

on mGerarTelasTeste(me,tLista)
    tMemberPergunta = ((tLista[1])[1])[1]

  tListaAlternativas = (tLista[1])[2]

  mGravarPerguntaAtual(script "quadroNegro",1)

  tListaPerguntaComRespostas = []
  tListaPerguntaComRespostas.append(tMemberPergunta)

  repeat with i = 1 to tListaAlternativas.count()

    tListaPerguntaComRespostas.append(tListaAlternativas[i])
  end repeat

  sendSprite(1,#mPosicionaItemTeste,tListaPerguntaComRespostas)
end mGerarTelasTeste
```

Se for selecionada a opção de formular exercício, aparecerá a tela de seleção, conforme demonstra a figura 6.4.5, onde o usuário poderá optar por elaborar exercício de quebra-cabeças. Na elaboração do quebra-cabeças, o professor seleciona uma imagem, que condiz

com o contexto exposto e grava esta imagem como arquivo em uma pasta que será acessada pelo aluno que executar o exercício.

**FIGURA 6.4.5 – TELA DE SELEÇÃODO MÓDULO FORMULAR EXERCÍCIO**


Quando o usuário selecionar o módulo aluno, será apresentada uma tela de seleção, conforme mostra a figura 6.4.6, onde terá duas opções, executar prova ou executar exercício. O aluno digita os seus dados, para que o protótipo possa recuperar e mostra quando o professor pedir para corrigir a prova. Independente da opção que ele for selecionar, o protótipo recupera o teste que já foi gerado anteriormente pelo professor.

**FIGURA 6.4.6 – TELA DE SELEÇÃO DE TAREFA DO MÓDULO ALUNO**


Se o aluno optar por executar prova, o protótipo executa um *behavior* para que o aluno abra a pasta e o arquivo onde está o teste designado pelo professor para ser executado. Após ter selecionado o arquivo, o protótipo mostra as telas para execução da prova, conforme mostra a figura 6.4.7. Sempre aparecerá uma questão com várias alternativas de respostas. O aluno clica na resposta que achar correta e pula para a próxima, até concluir o teste. Podendo se quiser, antes de concluir, voltar às questões anteriores.

**FIGURA 6.4.7 – TELA EXECUTAR PROVA**

O quadro 6.4.4 mostra parte do código da opção de executar prova. Foi implementado um teste para verificar a posição clicada e gravar esta posição, para que possa ser comparada com o gabarito e mostrar seu desempenho quando for solicitado a correção do teste. E também evitar que o aluno possa selecionar mais de uma opção. Quando o aluno concluir o teste, todas as posições clicadas e seus dados são gravados em um arquivo na pasta determinada.

**QUADRO 6.4.4 – PARTE DO CÓDIGO EXECUTAR PROVA**

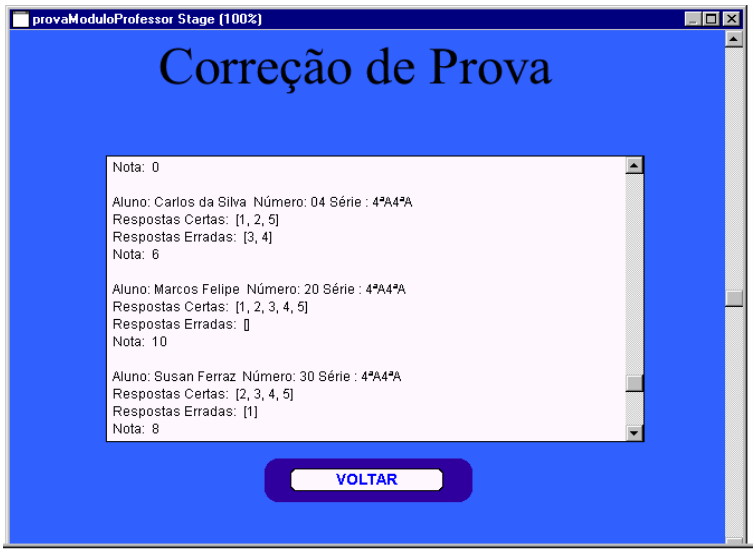
```
on mVerSeClicada me
  -- "lendo" nome do Cast Member com a pergunta para sabermos qual é o seu
número
  tMemberPergunta = Sprite(pPErguntaSprite).member.name
  tNumeroPergunta = tMemberPergunta.char[tMemberPergunta.char.count]
  pNumeroPergunta = value(tNumeroPergunta)
  tRespostaAnterior = gListaRespostas[pNumeroPErgunta]
  if tRespostaAnterior = 0 then
    member(pMeuMember).hilite = FALSE
  else
    if tRespostaAnterior = pMeuMember then
      member(pMeuMember).hilite = TRUE
    else
      member(pMeuMember).hilite = FALSE
    end if
  end if
end mVerSeClicada me
-- quando uma resposta for clicada
on mouseUp me
  -- criar lista dos sprites com resposta
  pListaSprites = value(pListaSprites)
  -- marcar a rsposta clicada
  member(pMeuMember).hilite = TRUE
  -- colocar resposta no recipiente (lista) para disponibilizá-la
  gListaRespostas[pNumeroPergunta] = pMeuMember

  repeat with i in pListaSprites
    if i <> pMeuSprite then
      sendSprite(i,#mDesmarcar)
    end if
    updateStage
  end repeat
end mouseUp me
-- handler para desmarcar resposta
on mDesmarcar me
  member(pMeuMember).hilite = FALSE
  updateStage
end mDesmarcar me
```

Depois que os alunos executaram a prova, o professor executa a opção de corrigir prova para obter o desempenho dos alunos. Ao selecionar a opção, o protótipo compara as respostas do aluno com o gabarito inserido pelo professor e calcula sua nota, mostrando em

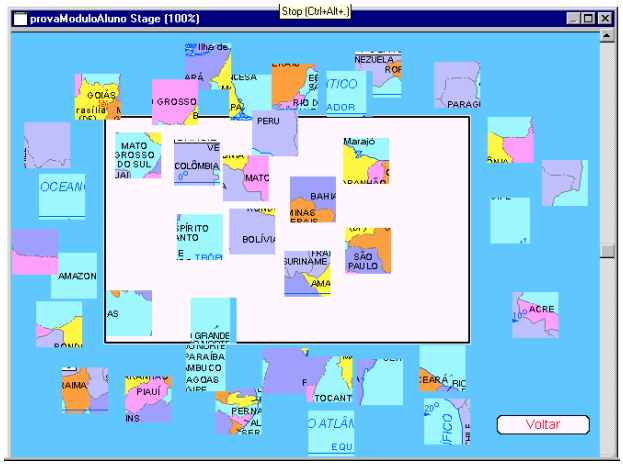
seguida a tela de correção, conforme mostra a figura 6.4.8, onde aparece o nome de cada aluno que executou a prova com seus dados, respostas certas, respostas erradas e respectiva nota.

**FIGURA 6.4.8 – TELA GERADA DA CORREÇÃO DE PROVA**




Se o usuário optar por executar exercício, aparecerá a tela para que o aluno execute o exercício de quebra-cabeças. Quando o aluno pedir para iniciar o exercício de quebra-cabeças, ele deverá selecionar a imagem determinada pelo professor. Após selecionada a imagem, o protótipo recorta esta imagem em peças de tamanhos iguais, e posiciona estas peças aleatoriamente na tela, junto com a área de montagem, conforme mostra a figura 6.4.9. Para executar o exercício, o aluno vai clicando e arrastando cada uma das peças para área de montagem, até formar a imagem correspondente. Cada peça colocada no lugar errado, o protótipo não a encaixará corretamente. O quadros 6.4.5 e 6.4.6 mostram respectivamente, parte do código em que são recortadas as peças e posicionadas no *stage*.

**FIGURA 6.4.9 – TELA DO EXERCÍCIO DE QUEBRA-CABEÇAS**




**QUADRO 6.4.5 – PARTE DO CÓDIGO DO BEHAVIOR PARA RECORTAR IMAGEM DO QUEBRA-CABEÇA**

```
on mCriarPecas(me)
  pListaPecas = [:]
  tTopPeca = 0
  tLinhas = pHeight / 50
  tColunas = pWidth / 50
  repeat with i = 1 to tLinhas
    tEsquerdaPeca = 0
    repeat with j = 1 to tColunas
      tRectLeft = 100 + tEsquerdaPeca
      tRectTop = 90 + tTopPeca
      tRectRight = tRectLeft + 50
      tRectBottom = tRectTop + 50
      if tRectRight > sprite(pMeuSprite).right then
        tRectRight = sprite(pMeuSprite).right
      end if
      if tRectBottom > sprite(pMeuSprite).bottom then
        tRectBottom = sprite(pMeuSprite).bottom
      end if
      tBasePeca = image(50,50,16)
      tMemberPeca = new(#bitmap,castLib "Pecas")
      tNomePeca = "Peca" & i & j
      tMemberPeca.image = tBasePeca
      tMemberPeca.name = tNomePeca
      tRectPeca = rect(tRectLeft,tRectTop,tRectRight,tRectBottom)
      put "Rect da peca" && tRectPEca
      tPosicaoCorreta = point((tRectLeft + 25),(tRectTop + 25))
      pListaPecas.addProp(symbol(tNomePeca),tPosicaoCorreta)
      tMemberPeca.image.copyPixels(the
stage.image,tMemberPeca.rect,tRectPeca)
      tEsquerdaPeca = tEsquerdaPeca + 50
    end repeat
   tTopPeca = tTopPeca + 50
 end repeat
end mCriarPecas
```

**QUADRO 6.4.6 – PARTE DO CÓDIGO DO BEHAVIOR PARA POSICIONAR PEÇAS NO STAGE**

```
on beginSprite(me)
  pMeuSprite = sprite(me.spriteNum)
  pMeuMember = sprite(pMeuSprite).member
end beginSprite
on mPosicionarAleatoriamente(me,tLista)
  tListaPecas = duplicate(tLista)
  if tListaPecas.count() > 1 then
    tNomeMember = tListaPecas.getPropAt(1)
    pPosicaoCorreta = tListaPecas[1]
    tListaPecas.deleteAt(1)
    tMinhaPosicaoH = random(400)
    tMinhaPosicaoV = random(400)
    sprite(pMeuSprite).member = member(string(tNomeMember),"Pecas")
    sprite(pMeuSprite).loc = point(tMinhaPosicaoH,tMinhaPosicaoV)
    updateStage

    tSprite = pMeuSprite + 1
    sendSprite(tSprite,#mPosicionarAleatoriamente,tListaPecas)
  end if

end mPosicionarAleatoriamente
```

## 6.5 TESTE E MANUTENÇÃO

Os testes realizados durante o desenvolvimento do trabalho possibilitaram melhorias no seu desempenho e a avaliação de novas possibilidades para a comodidade do usuário (professor).

O protótipo foi testado por 3 professores do ensino fundamental : professor de história, geografia e informática. São disciplinas que executam testes de múltipla escolha para avaliação do conteúdo exposto. A avaliação do desempenho do protótipo foi considerável, tendo sido levantadas algumas sugestões para melhorias, uma delas quanto a segurança ao acesso dos alunos ao módulo professor, que o protótipo não apresenta. Outras sugestões foram mencionadas mais adiante para trabalhos futuros. As opiniões foram semelhantes quanto a possibilidade de diminuir o trabalho na elaboração e correção de provas. Desconheciam a existência de  algum software que auxiliasse o professor neste aspecto, fazendo com que fizessem a elaboração e correção no modo tradicional. Despertou-se a curiosidade pela busca deste tipo de auxílio.

O protótipo não foi testado com alunos pelo fato dos alunos não terem acesso aos computadores no momento do desenvolvimento do protótipo. No módulo do aluno, os testes foram todos simulados.

# 7 CONCLUSÃO

O desenvolvimento deste protótipo demonstrou-se de grande viabilidade, pois possibilita ao professor dinamizar o processo de elaboração e correção, principalmente de provas, que dispende um enorme tempo. Processo que se torna complicado, principalmente, quando o professor possuir diversas turmas e precisar registrar avaliações. Partindo deste princípio, o professor pode dedicar mais tempo ao desenvolvimento e dinamização de suas aulas.

Durante a construção do protótipo, foram utilizadas algumas etapas da Metodologia OOHDM, metodologia utilizada para desenvolvimento de aplicações hipermídias. O software de autoria Macromedia Director 8.0 permitiu a realização do projeto de forma simples, pois possui grande variedade de recursos.

Os resultados obtidos com o protótipo atenderam aos objetivos propostos. Porém a possibilidade de se implementar melhoras no protótipo em futuras versões é válida, pois permite o aprimoramento de seu funcionamento.

## 7.1 SUGESTÕES PARA TRABALHOS FUTUROS

O uso de softwares educacionais que auxiliem no processo de avaliação eletrônica de alunos, com intuito de agilizar o desenvolvimento das atividades em sala de aula, é um passo importante no processo de ensino-aprendizagem, desenvolvendo também o raciocínio, motivação e criatividade dos alunos.

Entretanto, há a possibilidade de desenvolvimento de trabalhos de extensão. Apresenta-se aqui algumas sugestões:

f) desenvolver o software para internet, colocando-o na página do professor, para que os alunos acessem e executem os exercícios e provas via internet;
g) desenvolver o software on-line, onde o professor reúne um grupo de alunos interligados ao seu computador, elaborando as questões e os alunos respondendo on-line;

h) desenvolver software educativo que utilize agentes inteligentes para criar níveis de dificuldade nos exercícios propostos;
i) aperfeiçoar este protótipo, utilizando a inteligência artificial para gerar os exercícios, minimizando o trabalho de digitação do professor.

# REFERÊNCIAS BIBLIOGRÁFICAS

ANDRES, Daniela Pinto et al. Um estudo teórico sobre as técnicas de avaliação de software educacional. In: IX Seminário de Computação, 1., 2000, Blumenau. **Anais**.... CD, artigo 21.

BIZZOTTO, Carlos Eduardo Negrão. **Concepção, desenvolvimento e avaliação de um ambiente extensível para o aprendizado cooperativo distribuído**. Florianópolis, 1999. Projeto (Qualificação do Doutorado em Engenharia de Produção) Universidade Federal de Santa Catarina.

BIZZOTTO, Carlos Eduardo Negrão. **Director 8, rápido e fácil.** São Paulo: Makron Books, 2000.

CAMPOS, Gilda H. Bernardino de. **Metodologia para avaliação da qualidade de software educacional :**diretrizes para desenvolvedores e usuarios /Gilda Helena Bernardino de Campos. - 1994. - xii, 232p. :il.

CERQUEIRA, Alessandro de Almeida Castro – **Tese submetida para o grau de Mestre em Ciência em Engenharia de Sistemas e Computação** , Rio de Janeiro , mai. 1999. Disponível em: <http://WWW.nce.ufrj.br/~castro/tese/download.html>. Acesso em: 02/05/2001.

DALFOVO, Regiani. **Protótipo de software para o ensino de introdução à microinformática**. 1997. 65 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DETTMER, Brigida. **Protótipo de software para o ensino de informática básica**.1996.72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FURLAN, José Davi. **Modelagem de objetos através da UML**. Rio de Janeiro: Makron Books, 1998.

HUGO, Marcel. **Apostila programação orientada a objetos**, Blumenau, nov. 1999. Disponível em: <http://www.furb.rct-sc.br/~marcel/>. Acesso em: 07 maio 2001.

JUNIOR, Silvino Schlickmann. **Linguagens orientadas a objetos**, São Paulo, jun. 1999. Disponível em: <http://www.cfh.ufsc.br/~junior/linguagens.htm>. Acesso em: 04/04/2001.

LUCENA, Carlos; FUKS, Hugo. **A educação na era da internet**. Rio de Janeiro: Clube do Futuro, 2000.

MEC. Ministério da educação. Desenvolvido pelo Departamento de Informática de educação a distância, 2001. Apresenta textos sobre o Programa nacional de informática na educação – Proinfo. Disponível em : Disponível em: <http://www.proinfo.gov.br>. Acesso em: 30/04/2001.

POMPERMAIER, Leandro Bento. **Modelagem de aplicações hipermídia :** uma experiência com OOHDM, Rio grande do Sul, out. 2000. Disponível em: <http://www. inf.ufrgs.br/~pomper/oohdm.htm>. Acesso em: 09/05/2001.

ROCHA, Helder L. S. da. **Tutorial – programação orientada a objetos**, Rio de Janeiro, set. 2000. Disponível em: http://www.dsc.ufpb.br/~helder/java/Cap_2.html. Acesso em : 03/04/2001.

RUMBAUGH, James. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus, 1994.

SCHWABE, Daniel; ROSSI, Gustavo. **The object-oriented hypermedia design model-OOHDM**, Rio de Janeiro, out. 2000. Disponível em: <http://www.telemidia.puc-rio.br/oohdm/site_oohdm/oohdm.html>. Acesso em: 27 mar. 2001.

SMALL, Peter. **Lingo sorcery:** the magic of lists, objects and intelligent agents. Chichester : John Wiley, 1999.

TAJRA, Sanmya Feitosa. **Informática na educação:** professor na atualidade. São Paulo: Érica, 1998.

VALENTE, José Armando. **Questão do software :** parâmetros para o desenvolvimento do software educativo. Campinas, 1995. Núcleo de informática aplicada a educação. Universidade Estadual de Campinas.