UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE AMBIENTE MULTIUSUÁRIO PARA *CHAT* E PINTURA COLETIVA.

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

JÚLIO CÉSAR DE SOUZA

BLUMENAU, DEZEMBRO/2001

2001/2-30

PROTÓTIPO DE AMBIENTE MULTIUSUÁRIO PARA *CHAT* E PINTURA COLETIVA

JÚLIO CÉSAR DE SOUZA

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 N. Bizzotto — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Carlos Eduardo N. Bizzotto

_____

Prof. Dalton Solano dos Reis

_____

Prof. Dr. Oscar Dalfovo

DEDICATÓRIA

Dedico este trabalho a uma pessoa muito especial que é a minha esposa Mariana, que com carinho e sem medir esforços, soube me entender, apoiar e me ajudar para que eu pudesse chegar até aqui.

Dedico também a todos os meus familiares que compreenderam a minha ausência e sempre me apoiaram durante toda esta jornada, em especial ao meu pai (*in memoriam*) que de onde ele estiver , sei que está torcendo por mim.

"O degrau da escada não foi inventado para repouso, mas apenas para sustentar o pé o tempo necessário para que o homem coloque o outro pé, um pouco mais alto."

(RUXLEY)

AGRADECIMENTOS

Agradeço ao meu orientador, professor Carlos Eduardo Negrão Bizzotto pelo apoio e orientação deste trabalho.

Aos professores, Oscar Dalfovo e Wilson Pedro Carli que contribuíram com os seus conhecimentos na modelagem do protótipo.

Aos meus colegas de faculdade e amigos que de uma maneira direita ou indiretamente fizeram parte desta etapa.

E por fim, a todos aqueles que contribuíram de alguma forma para a conclusão deste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE QUADROS

RESUMO

O presente trabalho visa o estudo da função multiusuário do ambiente de autoria Macromedia Director 8.0 e o desenvolvimento de um protótipo de aplicação que inclui um *chat* que permite a utilização de texto e desenho. O protótipo desenvolvido pode ser utilizado como ferramenta auxiliar no processo de ensino-aprendizagem de disciplinas, pois permite a troca de informações através da Internet ou redes menores por intermédio do *chat* que dispõe as mensagens por texto e/ou por desenho. Para a elaboração do desenho o protótipo contém uma barra de ferramentas com diferentes pincéis e paleta de cores. A especificação do protótipo foi realizada utilizando a modelagem através de objetos no padrão UML.

ABSTRACT

The present work aims the study of the multiuser function of the environment by Macromedia Director 8 and the development of a prototype application that includes a chat that allows text and drawing. The developed prototype can be used as an auxiliary tool in the teaching-learning process of disciplines because it allows the exchange of information through the Internet or smaller networks through the chat which disposes messages by text and/or by drawing. For drawing elaboration, the prototype contains a toolbar with different brushes and colors palette. The prototype specification was accomplished through object modeling using UML standard.

1 INTRODUÇÃO

Os métodos de ensino nas escolas primárias, de nível médio e superior estão sofrendo grandes mudanças, principalmente no que diz respeito ao modo do professor interagir com os alunos (Grinkraut, 1996).

No ensino tradicional, o professor é o centro do processo, enquanto o aluno, é apenas um receptor de conteúdos reproduzindo somente aquilo que lhe foi instruído, não criando nada de novo com o que recebeu (Bizzotto, 1999). A personalização do atendimento é dificultada em função do número de alunos e da diversidade de interesses, habilidades e diferenças pessoais. As dificuldades para manter o interesse do aluno em sala de aula, e também o problema quanto a exposição do conteúdo programático pelo professor, estão presentes no dia-a-dia (Georg, 2000).

As novas tecnologias estão permitindo a prática de métodos pedagógicos revolucionários mais adequados às características e potencialidades da inteligência humana. A geração de ambientes interativos e assíncronos proporcionam conhecimentos infinitos e recursos derivados da microeletrônica digital, facilitam o processo de aprendizagem, pois permitem que o aluno assuma um papel ativo no processo e elimina o fator de distância e tempo (Wilhelm, 1997).

A introdução do computador na educação traz novas alternativas ao processo ensino-aprendizagem. A quantidade de programas educacionais e as diferentes modalidades de uso do computador mostram que esta tecnologia pode ser bastante útil (Santos, 1997), fazendo do aprendizado uma experiência mais cooperativa.

Diante disso, surgem os softwares hipermídia, que conforme Bizzotto (2000), pode ser considerados uma das grandes revoluções da mídia, uma vez que o aluno deixa de ser um simples usuário, passando a ser um participante ativo do ambiente criado pelo software.

É visando ampliar um pouco mais o conhecimento em relação a estes softwares, que este trabalho se propõe a desenvolver um protótipo de um ambiente multiusuário que permita a realização de *chat* e desenho coletivo "*on-line*", ou seja, um ambiente onde professor e alunos possam interagir entre si tendo ao seu dispor não só a comunicação por texto (*chat*),

mas também, podendo se utilizar de uma barra de ferramenta que disponibilize a criação de desenhos.

Na construção do protótipo foi utilizado a ferramenta Macromedia Director 8, onde o foco principal esta voltado para a sua função multiusuário, a qual foi estudada em termos de suas funcionalidades, facilidade de utilização e criação de aplicações.

1.1 OBJETIVOS

O presente Trabalho de Conclusão de Curso tem como objetivo o estudo da função multiusuário do ambiente de autoria Macromedia Director 8 e a implementação de um protótipo que permita a realização de *chat* que permita a inclusão de texto e desenho.

Os objetivos específicos do trabalho são:

a) estudar a função multiusuário do Macromedia Director 8.0.
b) permitir que os usuários participem de um *chat*, através da inclusão de texto e desenho.
c) possibilitar que o usuário utilize diferentes pincéis e cores para a realização do desenho.
d) indicar aos usuários as pessoas que estão participando do chat.
e) possibilitar a utilização do protótipo desenvolvido em um *browser*.

1.2 JUSTIFICATIVA

A partir do início dos anos 90, os recursos associados à multimídia passaram a ser integrados à rede das redes – a Internet. Esta é uma das mais inovadoras ferramentas de comunicação e informação, um canal para troca de idéias e experiências. É baseado nisto, que se define a necessidade de explorar as possibilidades de sua utilização como ferramenta educacional.

O uso da Internet nas escolas tem sido debatido em diferentes contextos. Contudo, tal como demonstram as iniciativas britânicas que buscam incorporar os avanços tecnológicos recentes ao ensino, é preciso ressaltar a necessidade de fazer com que as novas tecnologias sejam colocadas a serviço de projetos pedagógicos bem definidos.

Face a esta realidade, o presente trabalho visa explorar a capacidade multiusuário da ferramenta de autoria Director 8 para aplicações que permitam a interação "*on-line*" dos usuários.

1.3 ORGANIZAÇÃO DO TEXTO

O trabalho está estruturado em sete capítulos descritos a seguir:

O primeiro capítulo define os objetivos do trabalho, apresentando a justificativa para a sua elaboração.

O segundo capítulo fala sobre a informática na educação.

O terceiro capítulo aborda o tema interatividade e Internet com a apresentação de ferramentas destinadas a este fim.

O quarto capítulo apresenta o ambiente de desenvolvimento, ou seja, descreve a ferramenta Director 8 com ênfase para a sua função multiusuário.

O quinto capítulo aborda conceitos gerais aplicados a metodologia de orientação a objetos e a técnica de modelagem UML, usada na modelagem do protótipo.

O sexto capítulo aborda o desenvolvimento do protótipo, especificação e implementação.

O sétimo capítulo faz o fechamento deste trabalho apresentando as conclusões e sugestões para trabalhos futuros.

2 INFORMÁTICA NA EDUCAÇÃO

A aplicação da informática na educação tem se mostrado o foco de muitas pesquisas devido à importância da utilização de ferramentas computacionais vêm obtendo no apoio ao processo de ensino-aprendizagem.

No início os computadores eram vistos na área educacional, como meio de se aumentar a produtividade promovendo o ensino a um grande número de pessoas dentro de um curto espaço de tempo. Nos tempos atuais, o uso do computador na educação destaca-se pelo seu grande potencial no desenvolvimento das habilidades cognitivas essenciais para incrementar os processos de ensino, aprendizagem e treinamento (Silva, 2000).

Apesar de todos os recursos providos pelos computadores, ainda se encontra resistências em relação ao seu uso na educação em função de um processo tradicional de ensino ligado a velha forma de repassar o conhecimento, caracterizado basicamente por salas de aula, lousa e giz. Esse paradigma tem sido vencido a medida em que cresce a conscientização de que o computador é um instrumento de auxilio que facilita o acesso à informação e não pretende ser um substituto do professor (Silva, 2000).

2.1 HISTÓRICO

A primeira revolução tecnológica no aprendizado foi provocada por Comenius (1592-1670), quando transformou o livro impresso em ferramenta de ensino e de aprendizagem, com a invenção da cartilha e do livro texto. Sua idéia era utilizar esses instrumentos para viabilizar um novo currículo, voltado para a universalização do ensino.

Com o avanço tecnológico nasce o primeiro computador eletrônico programável, o *Electrical Numerical Integrator and Computer (ENIAC*), operacionalizado inicialmente em 1946, resultante de um esforço de guerra, necessário à participação militar americana na Segunda Guerra Mundial. Esse é um marco na história da computação, na resolução de problemas, apoiada em fundamentos lógicos. Datam igualmente da década de 40 os primórdios do lançamento da primeira etapa das assim chamadas “ciências da cognição”, envolvendo diversas disciplinas tais como a lingüística, a epistemologia, a psicologia cognitiva, as neurociências e, claro, a inteligência artificial (Meyer, 2000).

No final da década de 60 e início da década de 70 surgem os primeiros passos rumo ao ensino utilizando-se um computador. Algumas grandes empresas de computadores dos EUA, fizeram investimentos importantes nos denominados *Integrated Learning System (ILS)*, ou seja, Sistema Integrado de Aprendizagem, baseados em *mainframe*, e que trariam a instrução auxiliada por computador – *Computer Assisted Instruction (CAI)*, para as escolas em alta escala (Meyer, 2000).

Segundo Meyer (2000), o *PLATO (Programmed Logic for Automatic Teaching Operations)* da *Control Data Corporation* foi o primeiro sistema integrado de aprendizagem e também o maior fracasso em função do seu alto custo, acumulando uma perda de bilhões de dólares, o que resultou no seu abandono.

Com o aparecimento dos microcomputadores, no inicío dos anos 80, houve uma grande disseminação dos mesmos nas escolas. Essa conquista incentivou uma enorme produção e diversificação de *CAIs*, como tutoriais, programas de demonstração, exercício-e-prática, avaliação do aprendizado, jogos educacionais e simulação. De acordo com os estudos feitos pelo *The Educacional Products Information Exchange (EPIE) Institute* uma organização do *Teachers College*, da Universidade da Columbia, foram identificados em 1983 mais de 7.000 pacotes de software educacionais no mercado, sendo que 125 eram adicionados a cada mês. Isso aconteceu durante os primeiros três anos após a comercialização dos microcomputadores.

Entretanto, a presença dos microcomputadores permitiu também a divulgação de novas modalidades de uso do computador na educação como ferramenta no auxílio de resolução de problemas, na produção de textos, manipulação de bancos de dados e controle de processos em tempo real. De acordo com essa abordagem, o computador passou a assumir um papel fundamental de complementação, de aperfeiçoamento e de possível mudança na qualidade da educação, possibilitando a criação de ambientes de aprendizagem. O *Logo* foi o exemplo mais marcante dessa proposta (Meyer, 2000).

A linguagem *Logo* foi desenvolvida em 1967 tendo como base a teoria de Pieget e algumas idéias da Inteligência Artificial. Inicialmente essa linguagem foi implementada em computadores de médio e grande porte, fato que fez com que, até o surgimento dos microcomputadores, o uso do *Logo* ficasse restrito às universidades e laboratórios de pesquisa.

A proliferação dos microcomputadores no início da década de 90, permitiu o uso do computador em todos os níveis da educação americana. O computador é largamente utilizado na maioria das escolas de 1° e 2° graus e universidades.

Segundo Meyer (2000), entre 1984 e 1993, a porcentagem de alunos utilizando computadores nas escolas dos EUA, dobrou. As imagens gráficas dos PCs e as capacidades de som permitiram que os programadores criassem a multimídia *CAI*, em que som de alta qualidade, imagens gráficas e vídeo aprimoram a experiência de aprendizagem.

No Brasil, como em outros países, o uso do computador na educação teve início com algumas experiências em universidades, no inicio da década de 70. Na UFRJ, em 1973, o Núcleo de Tecnologia Educacional para a Saúde e o Centro Latino-Americano de Tecnologia Educacional (*NUTES / CLATES*) usou o computador no ensino de Química. Na UFRGS, nesse mesmo ano, realizaram-se algumas experiências usando simulação de fenômenos de física com alunos de graduação. O Centro de Processamento de Dados desenvolveu o software *SISCAI* para avaliação de alunos de pós-graduação em Educação. Na UNICAMP, em 1974, foi desenvolvido um software, tipo *CAI*, para o ensino dos fundamentos de programação da linguagem *BASIC*, usado com os alunos de pós-graduação em Educação, produzido pelo Instituto de Matemática, Estatística e Ciência da Computação e financiado pela Organização dos Estados Americanos.

É na realização do primeiro e segundo Seminário Nacional de Informática em Educação realizados respectivamente na Universidade de Brasília em 1981 e na Universidade Federal da Bahia em 1982, que a implantação da informática na educação no Brasil é implantada (Valente, 2001).

Atualmente com o crescimento explosivo da Internet, da Web e com a integração da informática as possibilidades se ampliaram. Muitos software utilizando-se desses meios já se encontram no auxílio à disciplinas e ao ensino das mesmas.

2.2 O COMPUTADOR COMO FERRAMENTA DE APOIO

Segundo Barker (1992), existem duas categorias referentes a aplicação dos computadores em ambientes de ensino:l

- aprendizado assistido por computador (*Computer Aided Learning – CAL*) que focaliza o uso dos computadores como ferramenta para promoção do aprendizado, visto como um agente que promove nova forma de transmissão dos conteúdos;
- treinamento baseado em computador (*Computer Basic Training – CBT*) que focaliza o computador como facilitador no aprendizado de tarefas específicas em um determinado domínio de conhecimento, inclusive com o uso de simulações.

Segundo Valente (1999), a introdução do computador em ambientes de ensino envolve basicamente quatro componentes: o computador, o software, o professor e o aluno. O relacionamento entre esses componentes é que determina a classificação dos sistemas educativos.

Dentre as várias formas de aplicação dos computadores na educação, Valente(1999) destaca:

- os sistemas tutoriais: responsáveis pela instrução do estudante baseado na realização de pequenos cursos. Promovem de maneira mais atrativa o processo de ensino-aprendizagem quando se utilizam de diversos tipos de mídias como sons, vídeos e imagens e com a integração de simuladores e jogos educacionais;
- os sistemas de exercício e prática: proporcionam ao estudante diversas formas de testar seu conhecimento através de exercícios e questionários, além de fornecer *feedback* imediato tanto para o estudante quanto para o professor;
- as ferramentas de manipulação de informação: compreendem uma grande quantidade de ferramentas específicas que podem ser utilizadas tanto pelo professor quanto pelo estudante, com os mais variados objetivos, tais como o processamento de texto, planilhas, manipulação de bancos de dados, construção e transformação de gráficos, sistemas de autoria, gerenciamento de informações, dentre outros.

2.3 TIPOS DE SOFTWARES EDUCACIONAIS

Dentro da concepção construtivista, um software para ser educativo deve ser um ambiente interativo que proporcione ao aluno investigar, levantar hipóteses, testá-las e refinar suas idéias iniciais. Dessa forma o aluno estará construindo o seu próprio conhecimento.

Para Valente (1998), a realização do ciclo descrição - execução - reflexão - depuração - descrição é de extrema importância na aquisição de novos conhecimentos por parte do aluno.

Os diversos tipos de softwares usados na educação podem ser classificados em algumas categorias, de acordo com seus objetivos pedagógicos: tutoriais, programação, aplicativos, exercício e práticas, multimídia e Internet, simulação e modelagem e jogos.

2.3.1 TUTORIAIS

Caracterizam-se por transmitir informações pedagogicamente organizadas, como se fossem um livro animado, um vídeo interativo ou um professor eletrônico. A informação é apresentada ao aluno seguindo uma seqüência pré-estabelecida.

A informação que está disponível para o aluno é definida e organizada previamente, assim o computador assume o papel de uma máquina de ensinar. A interação entre o aluno e o computador consiste na leitura da tela ou escuta da informação fornecida e avanço pelo material, apertando a tecla ENTER ou usando o mouse para escolher a informação.

Esse tipo de programa só permite ao professor verificar o produto final e não os processos utilizados para alcançá-lo. A sua limitação se encontra justamente no fato de não possibilitar verificar se “a informação processada passou a ser conhecimento agregado aos esquemas mentais", afirma Valente (1998).

2.3.2 EXERCÍCIOS E PRÁTICAS

Enfatizam a apresentação das lições ou exercícios. A ação do aluno se restringe a virar a página de um livro eletrônico ou realizar exercícios, cujo resultado pode ser avaliado pelo próprio computador. As atividades exigem apenas o fazer, o memorizar informação, não importando a compreensão do que se está fazendo (Valente, 1998).

2.3.3 PROGRAMAÇÃO

Esses softwares permitem que pessoas (professores ou alunos), criem seus próprios protótipos de programas, sem que tenham que possuir conhecimentos avançados de programação.

Ao programar o computador utilizando conceitos estratégias, este pode ser visto como uma ferramenta para resolver problemas. A realização de um programa exige que o usuário processe a informação, transformando-a em conhecimento.

Segundo Valente (1998), a programação permite a realização do ciclo descrição - execução - reflexão - depuração - descrição. O programa representa a idéia do aluno e existe uma correspondência direta entre cada comando e o comportamento do computador. As características disponíveis no processo de programação ajudam o aluno a encontrar seus erros, e o professor a compreender o processo pelo qual o aluno construiu conceitos e estratégias envolvidas no programa.

2.3.4 APLICATIVOS

São programas voltados para aplicações específicas, como processadores de texto, planilhas eletrônicas e gerenciadores de banco de dados. Embora não tenham sido desenvolvidos para uso educacional, permitem usos interessantes em diferentes ramos do conhecimento.

Valente(1998) defende que, nos processadores de textos, as ações do aluno podem ser analisadas em termos do ciclo descrição - execução - reflexão - depuração - descrição. Quando o aluno está digitando um texto no processador de texto, a interação com o computador é mediada pelo idioma materno e pelos comandos de formatação. Apesar de simples de serem usados e de facilitar a expressão do pensamento, o processador de texto não pode executar o conteúdo do mesmo e apresentar um feedback do conteúdo e do seu significado para o aluno. A única possibilidade, em se tratando de reflexão, é comparar as idéias originais do formato com o resultado apresentado, não dando margem para a reflexão e depuração do conteúdo. Nesse sentido, o processador de textos não dispõe de características que auxiliam o processo de construção do conhecimento e a compreensão das idéias.

2.3.5 MULTIMÍDIA E INTERNET

Em relação à multimídia, Valente (1998) chama a atenção para a diferenciação entre o uso de uma multimídia já pronta e o uso de **sistemas de autoria** para o aluno desenvolver sua multimídia.

Na primeira situação, o uso de multimídia é semelhante ao tutorial. Apesar de oferecer muitas possibilidades de combinações com textos, imagens, sons, a ação do aluno se resume em escolher opções oferecidas pelo software. Após a escolha, o computador apresenta a informação disponível e o aluno pode refletir sobre a mesma. Às vezes, o software pode oferecer também ao aluno oportunidade de selecionar outras opções e navegar entre elas. Essa idéia pode manter o aluno ocupado por um certo tempo e não lhe oferecer oportunidade de compreender e aplicar de modo significativo as informações selecionadas.

Dessa forma, o uso de multimídia pronta e Internet são atividades que auxiliam o aluno a adquirir informações, mas não a compreender ou construir conhecimentos com a informação obtida. Torna-se necessária a intervenção do professor para que o conhecimento seja construído.

Na segunda situação, sistemas de autoria, o aluno seleciona as informações em diferentes fontes e programas, construindo, assim, um sistema de multimídia. Dessa forma, é possibilitado ao aluno refletir sobre os resultados obtidos, compará-los às suas idéias iniciais e depurar em termos de qualidade, profundidade e significado da informação apresentada. Assim, pode-se garantir a realização do ciclo descrição - execução - reflexão - depuração - descrição para representar a informação de forma coerente e significativa.

O tipo de execução do sistema de autoria se assemelha ao processador de texto, pois executa uma sucessão de informação e não a própria informação; ele também não registra o processo que o aluno usa para montar o software multimídia.

2.3.6 SIMULAÇÃO E MODELAGEM

Constituem o ponto forte do computador na escola, pois possibilitam a vivência de situações difíceis ou até perigosas de serem reproduzidas em aula, permitindo desde a

realização de experiências químicas ou de balística, dissecação de cadáveres, até a criação de planetas e viagens na história.

A simulação pode ser de três tipos (Valente, 1998):

- Fechada - quando o fenômeno é previamente implementado no computador, não exigindo que o aluno desenvolva suas hipóteses, teste-as, análise os resultados e refine seus conceitos. Nessa perspectiva a simulação se aproxima muito do tutorial.
- Aberta - quando fornece algumas situações previamente definidas e encoraja o aluno a elaborar suas hipóteses que deverão ser validadas por intermédio do processo de simulação no computador. Neste caso, o computador permite a elaboração do nível de compreensão por meio do ciclo descrição - execução - reflexão - depuração - descrição, em que o aluno define e descreve o fenômeno em estudo.
- Modelagem - o modelo do fenômeno é criado pelo aluno que utiliza recursos de um sistema computacional para implementar esse modelo no computador, utilizando-o como se fosse uma simulação. Esse tipo de software exige um certo grau de envolvimento na definição e representação computacional do fenômeno e, portanto, cria uma situação bastante semelhante à atividade de programação, possibilitando a realização do ciclo descrição - execução - reflexão - depuração - descrição.

Para Valente (1998), a diferença entre simulação fechada, aberta, modelagem e programação está no nível de descrição que o sistema permite. Na programação, o aluno pode implementar o fenômeno que desejar, dependendo somente da linguagem de programação que for utilizada. Na modelagem, a descrição é limitada pelo sistema fornecido e pode-se restringir a uma série de fenômenos de um mesmo tipo. Na simulação aberta, o fenômeno pode estar definido e o aluno deverá implementar as leis e definir os parâmetros envolvidos. Na simulação fechada, a descrição se limita a definição dos valores de alguns parâmetros do fenômeno.

Portanto, para que a aprendizagem se processe é necessário que se propicie um ambiente onde o aluno se envolva com o fenômeno e o experencie, levantando suas hipóteses, buscando outras fontes de informação e usando o computador para validar sua compreensão

do fenômeno. A intervenção do "agente de aprendizagem" será no sentido de não deixar que o aluno acredite que o mundo real pode ser simplificado e controlado da mesma maneira que os programas de simulação, e de possibilitar a transição entre a simulação e o fenômeno no mundo real porque a mesma não é automática.

2.3.7 JOGOS

Geralmente são desenvolvidos com a finalidade de desafiar e motivar o aluno, envolvendo-o em uma competição com a máquina e os colegas. Os jogos permitem interessantes usos educacionais, principalmente se integrados a outras atividades.

Os jogos podem também ser analisados do ponto de vista do ciclo descrição - execução - reflexão - depuração - descrição, dependendo da ação do aluno em descrever suas idéias para o computador.

Valente (1998) alerta que os jogos têm a função de envolver o aluno em uma competição e essa competição pode dificultar o processo de aprendizagem uma vez que, enquanto estiver jogando, o interesse do aluno está voltado para ganhar o jogo e não em refletir sobre os processos e estratégias envolvidos no mesmo. Sem essa consciência é difícil uma transformação dos esquemas de ação em operação.

2.4 EXEMPLOS DE SOFTWARE EDUCACIONAIS

Os construtivistas estão vendo a educação baseada em computador de uma nova maneira. Em vez de ver os computadores como um meio de suplementar a instrução tradicional, eles os vêem como um meio de tirar os professores do modo expositivo. As informações ricas de softwares abertos podem fornecer uma maneira de mover o foco da sala de aula para atividades baseadas em projetos (Meyer, 2000).

2.4.1 ARCHAEOTYPE

Promovido pela Dalton School, uma escola particular de Nova York, o *Archaeotype*, é um software de simulação baseado em computador de uma escavação arqueológica na Assíria.

O objetivo deste software é fazer com que os alunos realizem escavações simuladas e através dos artefatos que descobrem e deduzem sobre a sociedade assíria. Quando os alunos escolhem as ferramentas dos menus do *Archaeotype* e escavam os artefatos, o aplicativo simula os sons feitos por ferramentas arqueológicas reais e revela os artefatos ocultos. A medida que os alunos tentam entender o que descobriram e fazem deduções sobre a sociedade e a cultura assírias, podem utilizar as bibliotecas *on-line* do *Archaeotype*, ricas em informações relacionadas com a Assíria, abordando assuntos como cerâmica, armaduras, arquitetura e transportes. Para terminar a sua interpretação, os alunos descobrem que devem avançar além dos estudos sociais e utilizar ferramentas fornecidas pela matemática, ciências, história e filosofia. Os alunos também são encorajados a fazer um esquema no computador. Eles consultam especialistas locais, assim como os incríveis recursos do Metropolitan Museum of Art (Meyer, 2000).

Uma das características-chave do *Archaeotype* é a imprevisibilidade: é completamente incerto como uma equipe de alunos irá explorar e interpretar o que eles encontraram. Essa incerteza, entretanto, é a questão. As sociedades e culturas do passado nunca puderam ser conhecidas com exatidão. Os arqueólogos discordam sobre suas interpretações da sociedade assíria e assim também o fazem os alunos. Além de aprenderem o que é dado como certo sobre a Assíria, os alunos aprendem a basear suas formulações sobre evidências efetivas, a utilizar ferramentas interdisciplinares para interpretar e comunicar suas interpretações de maneira que os outros a considerem convincente (Meyer, 2000).

2.4.2 WEB-QUEST

Criado por Bernie Dodge e Tom March em SanDiego State University, o *Web-Quest* é uma investigação orientada para desenvolver projetos baseados em computador, na qual algumas ou todas as informações com as quais os aprendizes interagem são originadas de recursos da Internet. O *Web-Quest* enfatiza pesquisas colaborativas nas quais os alunos trabalham em equipe para procurar informações e dar sentido às que encontram (Meyer, 2000).

Desenvolvidos por professores conectados na Internet, os módulos do *Web-Quest* começam com objetivos curriculares claramente estabelecidos, fornece estruturas bem

definidas para as pesquisas dos alunos e oferece orientação ao longo do caminho. Como o *Archaeotype*, os módulos do *Web-Quest* desafiam os alunos a descobrir, interpretar e comunicar conhecimento em vez de recebê-lo passivamente.

Um exemplo de módulo é de *Hello Dolly*, um *Web Quest* que explora as questões levantadas pela clonagem. Dolly é o nome da ovelha produzida pelo primeiro esforço bem-sucedido de clonagem de um animal. Projetado para alunos de sugundo grau, *Hello Dolly* incentiva os alunos a imaginarem-se como participantes em uma audiência no Senado dos EUA. Os alunos dividem-se em equipes para represenar os vários eleitorados afetados pela legislação proposta que proibiu a clonagem. Esses eleitorados incluem o Departamento da Agricultura dos EUA, pesquisadores de clonagem, um grupo de advogados de direitos dos animais, professores de ética biomédica e teólogos. Cada eleitorado tem seu próprio dossiê, que inclui perguntas iniciais, um plano de ação, conselho de apresentação e *links* úteis da Internet. Cada grupo faz sua apresentação, e um grupo de "Senadores" vota a lei (Meyer, 2000).

Graças à Internet, o *Hello Dolly* pode ser desenvolvido por uma fração minúscula do custo do *Archaeotype*. Em vez de desenvolver o próprio conteúdo, Nuthall, o autor, pode obter materiais ricos da própria Internet relativos à clonagem aos cientistas que clonaram, as opiniões contraditórias de vários eleitorados afetados, os materiais científicos na biologia de clonagem, às perspectivas teológicas sobre as tecnologias biológicas avançadas e muito mais. À medida que educadores criam e compartilham materiais educacionais ricos, como Hello Dolly, na Internet, mesmo distritos escolares mais pobres terão uma chance de experimentar os benefícios da reforma educacional (Meyer, 2000).

3 INTERATIVIDADE

Entende-se por interatividade a ação recíproca de dois ou mais corpos, uns nos outros. Em outras palavras, a interatividade está ligada à capacidade de um usuário de dar um rumo, a uma dada situação, de acordo com suas ações e decisões. Um meio, início e fim dinâmicos e variáveis. A interatividade transforma o usuário de um simples espectador para um elemento ativo, contrário a televisão, por exemplo, em que a pessoa não passa de um simples receptor do conteúdo apresentado.

A interatividade vem sendo usada em larga escala e seu crescimento é proporcional ao da Internet. Esse crescimento pode ser entendido melhor verificando ao longo do tempo a sua evolução.

A idéia de interatividade começou a nascer na área da literatura durante o movimento do modernismo que tinha como um dos seus objetivos, mudar os antigos valores do parnasianismo, considerados metódicos, ultrapassados e intransigentes. Uma das formas encontradas pelos primeiros poetas do movimento foi a quebra da linha seqüencial que acompanhava os poemas da época. Durante este período surge o livro-cubo que apresentava ao leitor uma história com vida independente, que presumia a investigação e levava em conta o raciocínio de quem vivenciava a obra (Mohandas, 2001).

Após esta fase, veio o concretismo trazendo consigo os livros-aventura que traziam um trecho de estória até que coloca várias alternativas de seqüência . De acordo com a resposta, o leitor era conduzido para outras páginas e assim sucessivamente até o seu fim. A estória ganhava diversos finais e seqüências aleatórias, caracterizando o uso da interatividade (Mohandas, 2001).

Avançando bastante no tempo, surgiram os PCs e com sua popularização, surgem as interfaces gráficas, como o Windows, que substitui os prompts de comandos que necessitavam de conhecimento técnico para utiliza-los com eficiência. Na seqüência, aparecem os primeiros disquetes e CD-ROM com programas multimídia, especialmente voltados para a educação.

Segundo Mohandas (2001), foi apartir dos anos 90 que houve um crescimento e popularização da multimídia e o surgimento das ferramentas visuais responsáveis pelo desenvolvimento de softwares multimídias.

Conforme Bizzotto (2000), a interação através de softwares multimídias era feita através de hipertextos onde o usuário clicava em uma palavra resultando no desvio para outro texto que, em geral, continha uma explicação detalhada. Com a evolução da tecnologia, surge o conceito de hipermídia que possibilita não só a interação em textos mas também com mídias como imagens e vídeos e com o interligamento das mesmas pode-se aumentar sensivelmente as possibilidades oferecidas pelos softwares hipertextos. A hipermídia pode ser considerada a maior revolução dos últimos tempos, passando o usuário a ser um elemento ativo do ambiente criado pelo software.

Juntamente com o crescimento de softwares hipermídia, a Internet toma dimensões grandiosas, a sofisticação cada vez maior dos navegadores atuais e as técnicas para a criação de experiências interativas tornou possível a criação de aplicativos mais funcionais (Szeto, 1997).

3.1 INTERNET

A Internet é uma rede de computadores interligados, mas independentes. Em menos de duas décadas, transformou-se de uma rede altamente especializada de comunicações, utilizada inicialmente para fins militares, acadêmicos e comerciais. Seu histórico e evolução podem ser acompanhados na tabela abaixo (Quadro 1).

QUADRO 1 – A HISTÓRIA DA INTERNET.

FASE	ANO	EVOLUÇÃO
INICIAL	1966	Um pesquisador da Agência de Projetos Avançados de Pesquisa (Arpa) chamado Bob Taylor consegue US$ 1 milhão para tocar um projeto de interligação dos laboratórios universitários que colaboram com a agência. O objetivo é economizar dinheiro ao compartilhar os recursos de computação espalhados pelo país.

Fonte: Adaptado de Mundo Digital(1997)

QUADRO 1 – A HISTÓRIA DA INTERNET(cont.).

FASE	ANO	EVOLUÇÃO
INICIAL	1967	Taylor convence Larry Roberts a trabalhar no projeto. Roberts, considerado a única pessoa nos Estados Unidos capaz de montar uma rede do gênero, faz o desenho da configuração original, interligando quatro centro de computadores.
	1968	Com o projeto aprovado, a Arpa abre licitação. Dezenas de empresas se candidataram. A IBM não participou, alegando que uma rede do gênero jamais poderia ser construída. A Bolt, Beranek and Newman (BBN) ganha a concorrência.
	1969	Em 1° de maio de 1969, a BBN envia o primeiro equipamento da rede para a Universidade da Califórnia em Los Angeles. A UCLA se tornaria o primeiro nó (ponto de conexão) da rede que viria a se chamar Arpanet e, mais tarde, Internet. Entre a equipe que a ajudou a montar o equipamento estava o então estudante de pós-graduação Vint Cerf, que mais tarde se tornaria presidente da Internet Society e vice-presidente da MCI, gigante do ramo de telecomunicações. No Instituto de Pesquisas de Stanford, Douglas Englebart montou o segundo nó da rede. Englebart é também o inventor do mouse. Até o final de 1969, mais dois centros de pesquisas foram conectados: Universidade da Califórnia em Santa Bárbara (UCSB) e Universidade de Utah.
	1971	A Arpanet chega a 15 nós com a inclusão de computadores da BBN, MIT, RAND Corporation, Universidade de Harvard, Universidade de Stanford, Universidade de Illinois em Urbana, Universidade Carnegie Mellon (CMU) e do centro de pesquisas Ames da Agência Nacional de Administração Espacial (NASA), entre outros.
	1973	São montadas as primeiras conexões internacionais com a Arpanet na University College de Londres e Royal Radar Establishment, na Noruega.
	1974	A BBN inaugura o Telenet, o primeiro serviço comercial conectado à Arpanet.
	1979	Criação da Usenet, a rede de grupos de discussão, com os grupos da hierarquia `net`. Na Universidade de Essex Richard Bartle e Roy Trubshaw lançam o primeiro MUD, uma mistura de canal de conversa em tempo real com RPG (Role Playing Game).
	1981	Uma rede cooperativa, chamada de Bitnet (Because It's Time NETwork), inicia na City University, de Nova York, oferecendo correio eletrônico, servidores de listas e transferência de arquivo. A Bitnet se torna uma alternativa à Internet.

Fonte: Adaptado de Mundo Digital (1997)

QUADRO 1 – A HISTÓRIA DA INTERNET(cont.).

FASE	ANO	EVOLUÇÃO
EXPANSÃO ACADÊMICA	1982	O nome Internet começa a ser utilizado para designar as redes que utilizam o conjunto de protocolos TCP/IP (Transmission Control Protocol/Internet Protocol), escritos por Vint Cerf e Bob Kahn, dois dos cientistas que participaram da montagem dos primeiros nós da Arpanet, em 1969. Holanda, Dinamarca e Suécia entram na rede.
	1983	Mai uma rede alternativa à Arpanet, a Earn (rede acadêmica européia) começa a funcionar. O financiamento é da IBM.
	1984	O número de servidores na rede chega a 1.000. O Japão cria o Japan Unix Network.
	1986	A NSFNet cria um canal de alta velocidade (para a época, 56Kbps) para conectar cinco centros de supercomputação. O resultado é uma explosão no número de universidades conectadas. O primeiro Freenet, serviço gratuito de acesso à rede, é criado no estado de Cleveland.
	1987	Número de servidores na Internet chega a 10.000, enquanto na Bitnet o número chega a 1.000.
	1988	Um programa clandestino perde o controle e afeta o funcionamento de 6.000 dos 60.000 servidores da rede. O vírus fica conhecido como Internet Worm (o Verme da Internet). Jarkko Oikarinen cria o Internet Relay Chat (IRC), um serviço de bate-papo pela Internet. Canadá, Dinamarca, Finlândia, França, Islândia e Noruega se conectam à rede.
	1989	Número de servidores chega a 100.000. Austrália, Alemanha, Israel, Itália, México, Nova Zelândia e Porto Rico se ligam à rede.
	1990	A Arpanet deixa de existir. Argentina, Áustria, Bélgica, Brasil, Chile, Grécia, Índia, Irlanda, Coréia do Sul Espanha e Suíca entram para a Internet.
	1991	O Gopher, um sistema de organização da informação na Internet na forma de menus e bancos de dados, é inventado por Paul Lindner e Mark P. McCahill da Universidade de Minnessota. Philip Zimmerman inventa um programa que permite enviar mensagens codificadas pela Internet, conhecido como PGP (Pretty Good Privacy). Mais países conectados: Croácia, República Tcheca, Hong Kong, Hungria, Polônia, Portugal, Singapura, África do Sul, Tailândia e Tunísia.
	1992	Fundação da Internet Society (Isoc). O Laboratório Europeu de Física de Partículas (Cern) inventa a World Wide Web (WWW). Criada inicialmente como uma ferramenta de trabalho para cientistas espalhados pelo mundo, a Web começa a ser utilizada para colocar informações ao alcance de qualquer usuário da Internet. O número de servidores na rede chega a 1 milhão. Países conectados: Camarões, Chipre, Equador, Estônia, Kuwait, Luxemburgo, Eslováquia, Eslovênia, Tailândia e Venezuela.

Fonte: Adaptado de Mundo Digital (1997)

QUADRO 1 – A HISTÓRIA DA INTERNET(cont.).

FASE	ANO	EVOLUÇÃO
	1993	O Centro Nacional de Aplicações de Supercomputação dos Estados Unidos (NCSA) lança o Mosaic. A Fundação Nacional de Ciência americana cria o InterNIC para organizar o registro de domínios (parte dos nomes dos computadores) e informações sobre a rede. O crescimento anual da WWW alcança 341.634%. Mais países conectados à Internet: Bulgária, Costa Rica, Egito, Fiji, Gana, Guam, Indonésia, Cazaquistão, Quênia, Liechtenstein, Peru, Romênia, Federação Russa, Turquia, Ucrânia e Ilhas Virgens.
EXPLOSÃO COMERCIAL	1994	O tráfego na NSFNet ultrapassa 10 trilhões de bytes por mês (o equivalente à capacidade de 16 mil discos de CD-ROM). First Virtual, o primeiro banco na Internet começa a funcionar. Algéria, Armênia, Bermuda, Burkina Faso, China, Colômbia, Polinésia Francesa, Líbano, Lituânia, Macau, Marrocos, Nova Caledônica, Nicarágua, Nigéria, Panamá, Filipinas, Senegal, Sri Lanka, Suazilândia, Uruguai e Uzbequistão entram para a rede.
	1995	A NSFNet volta ser uma rede exclusivamente acadêmica. O tráfego comercial nos Estados Unidos fica com a iniciativa privada. Serviços *on-line* tradicionais começam a oferecer acesso à Internet. As empresas criadas em torno da Internet vendem ações no mercado americano. As ações da Netscape, fabricante do navegador Netscape Navigator, alcançam valorização recorde.
	1996	O ano começa conturbado com o desligamento de grupos de discussão da CompuServe na Alemanha a pedido do governo para impedir a distribuição principalmente de pornografia. Nos Estados Unidos, o presidente Bill Clinton aprova uma nova lei de telecomunicações, que, entre outras resoluções, prevê pena para quem distribuir conteúdo considerado inadequado na Internet.Yahoo, Excite eLycos, três grandes serviços de busca na rede, lançam suas ações no mercado.

Fonte: Adaptado de Mundo Digital (1997)

Para se ter idéia do crescimento da Internet, entre os anos 1993 a 1998 o número de pessoas novas se conectando, dobrou todos os anos segundo Heide (2000). Qualquer pessoa que se conecta na Rede, logo descobre que ela é um microcosmo da nossa sociedade.

De acordo com Mohandas (2001) é comum confundir Internet com a WWW (*Wordl Wide Web*). A Internet agrega uma multiplicidade de serviços que estrapolam as páginas html com e-mail e protocólos de transferência de arquivos FTP (*File Trnasfer Protocol*). A WWW criada por Jim Beners Lee, corresponde a parte multimídia da Internet compreendendo os sites. Somente a partir da implantação da WWW é que a internet se popularizou.

A Internet, segundo Heide (2000), ou mais precisamente, o mundo dos computadores e das telecomunicações da qual a Internet é apenas uma parte, está reestruturando a sociedade. Através de um navegador da Web (como o Netscape, o Opera ou o Internet Explorer, por exemplo) como ferramenta de comunicação é possível enviar correio eletrônico ou participar de discussões on-line. Tecnologias como conferência na Web, em que as mensagens são publicadas ao vivo e a tecnologia *push*, que permite distribuição personalizada de dados, têm aprimorado a apresentação relativamente estática de texto e imagens gráficas.

3.2 EXEMPLOS DE SISTEMAS DE DESENVOLVIMENTO HIPERMÍDIA

A utilização de sistemas hipermídia para desenvolvimento de aplicações na área educacional tem contribuído para o aumento da qualidade e eficiência no aprendizado.

Serão apresentados seis sistemas de uma forma bastante genérica, mas em função de sua contribuição para o campo de desenvolvimento hipermídia, merecem uma atenção especial.

3.2.1 KMS

O KMS (*Knowledge Management System*) foi desenvolvido na Carnegie Mellon University projetado para gerenciar grandes redes de hipertexto através de redes locais.

Sua estrutura básica é o frame, que pode conter texto, gráficos e imagens. Os frames conectados a outros frames via ligações que podem ser de dois tipos: itens de árvore para representar relações hierárquicas e itens de anotação para representar relações referenciais. Não existe no KMS distinção entre os modos *browsing* e autoria. O usuário pode alterar um frame ou criar ligações à qualquer momento e as modificações são salvas automaticamente. Assim sendo o KMS pode ser uma valiosa ferramenta de autoria e de colaboração em uma rede multiusuário por não fazer distinção entre o desenvolvedor e o usuário.

Com essas características, esta ferramenta tem sido utilizada em trabalhos cooperativos, publicações eletrônicas, gerenciamento de projetos, manuais e mensagens eletrônicas Chaiben (1999).

3.2.2 INTERMEDIA

Originário da Brown University, esta ferramenta inicialmente era um projeto de pesquisa para auxiliar na melhoria do ensino e da pesquisa. Por ser um ambiente integrado, permite o compartilhamento num único ambiente hipermídia diferentes tipos de aplicações como processadores de texto, editores, etc. O Intermedia suporta grandes redes hipermídias com facilidades de navegação e *browsing*, incluindo redes semânticas e mapas conceituais. Alunos e professores tem se utilizado das facilidades de autoria para anotação e organização de materiais, criando ligações entre texto, diagramas, imagens, seqüências de vídeo, e som.

Desenvolvido originalmente sobre o sistema operacional UNIX e então migrando para o Macintosh II, esta é uma ferramenta tanto de autoria como de leitura para trabalhos escolares (Chaiben, 1999).

3.2.3 GUIDE

Desenvolvido como um projeto de pesquisa na University of Canterbury e comercialmente distribuído pela Owl Systems, Inc., o GUIDE pode ser utilizado tanto na plataforma IBM PC quanto para Apple Macintosh. Assim como no KMS, esta ferramenta não distingue entre os níveis de “autor” e “leitor”.

O GUIDE suporta quatro diferentes tipos de ligações:

- substituição - que troca o texto na janela de trabalho corrente por outro.
- notas - que chamam um texto subordinado em uma janela sobreposta.
- referências - que abrem uma janela de texto inteiramente nova e novos arquivos de texto se necessário.
- comandos - que mostram outras opções de ligações de substituição enquanto continua trabalhando dentro da janela de texto corrente.

Em 1990, é lançada a Versão 3.0, que inclui capacidades de pesquisa de texto entre arquivos, tecnologia de gerenciamento de objetos, e Extalk, uma extensão de linguagem de programação que permite melhor criação e controle da estrutura hipermídia (Chaiben, 1999).

3.2.4 NOTECARDS

Desenvolvido pela XEROX Palo Alto Research Center na linguagem de programação LISP. É um sistema para documentação, que opera com o conceito de "*cards*". Os "*cards*" são criados para vários propósitos e mostrados na tela. Sendo principalmente uma ferramenta de organização da informação, o sistema conta com uma rede de "*cards*" relacionados semanticamente e inclui ferramentas para a organização, gerenciamento, e exposição desta informação relacionada. Isto permite que um simples documento hipertexto apresente várias visões, dependendo de quais ligações estão correntemente prontas para serem exibidas. Adicionalmente o NoteCards produz "mapas de navegação" dos "*cards*" que estão associados com cada documento, e é possível saltar diretamente para qualquer "*card*", selecionando a parte apropriada do mapa de navegação.

É considerado um marco importante, pois ajudou a definir técnicas de navegação através de *browsing* com interfaces gráficas necessárias na elaboração de aplicações. O NoteCards demonstra também como as redes semânticas e hierarquias podem fundir para criar um sistema de recuperação e gerenciamento de informações altamente flexível e eficaz, assim como um esquema para a representação do conhecimento (Chaiben, 1999).

3.2.5 HYPERCARD

Tendo sido inspirado no NoteCard, esta ferramenta era inicialmente um pacote de aplicações criado para a Apple Macintosh. Surgiram seus similares como o MetaCard for X-Windows baseado em Unix, o SuperCard para Macintosh e o ToolBook para PC. Estas ferramentas possuem como característica a fácil manipulação das capacidades de interface gráfica e a unidade básica de armazenamento de texto é o card.

Outra característica é a linguagem *script* associadas permitindo a programação de aplicações complexas. No HyperCard a linguagem script é o Hyper Talk.

Esta ferramenta é de fácil integração com várias ferramentas de sistemas especialistas disponíveis para ambiente Macintosh, servindo de padrão e/ou modelo no desenvolvimento de trabalhos nas áreas de projetos de sistemas inteligentes e de projetos educacionais. A terminologia adotada pela Apple para descrever os componentes do HyperCard é seguido por outros sistemas hipermídia (Chaiben, 1999).

3.2.6 KNOWLEDGEPRO

Esta ferramenta combina as tecnologias de sistemas especialistas e hipertexto em uma única ferramenta de desenvolvimento para plataforma PC, com versão para o ambiente Windows. Desenvolvido pela Knowledge Garden, Inc., esta ferramenta de comunicação permite aos PCs tornarem-se um meio de comunicação de idéias e informações.

Considerada uma linguagem de alto nível que facilita o desenvolvimento de sistemas inteligentes utilizando uma variedade de estratégias. Ela pode representar o conhecimento procedimental como uma coleção de regras ou um conjunto de “tópicos” (a unidade de estrutura básica da linguagem). Um tópico pode conter comandos (como procedimentos), valores armazenados (como variáveis), valores de retorno (como funções), assinalar propriedades (como um “frame”), ser arranjado hierarquicamente e portanto demonstrar herança, comportar como comandos do sistema, ou até mesmo possibilitar as características de hipertexto (Chaiben, 1999).

O KnowledgePro não é uma ferramenta orientada à menu. Como uma linguagem compilada de alto nível, ela contém mais de 100 palavras chave que agem como comandos na linguagem, e rigorosas regras de sintaxe. A linguagem contém um rico conjunto de funções para manipulação de “strings” (parecido com LISP), que são bastante convenientes para o desenvolvimento de projetos sofisticados. O sistema não incorpora nenhuma ferramenta de auxílio para a navegação. Entretanto, os desenvolvedores podem explorar todos os recursos existentes num ambiente de desenvolvimento Windows, para criar seus próprios esquemas (Chaiben, 1999).

3.2.7 DIRECTOR

O Director é o mais popular representante dessa safra de softwares e é utilizado principalmente para fazer sistemas de menu, apresentações e jogos bidimencionais. Por se tratar do ambiente utilizado no desenvolvimento do protótipo, será descrito em maiores detalhes no próximo capítulo.

4 AMBIENTE DE DESENVOLVIMENTO

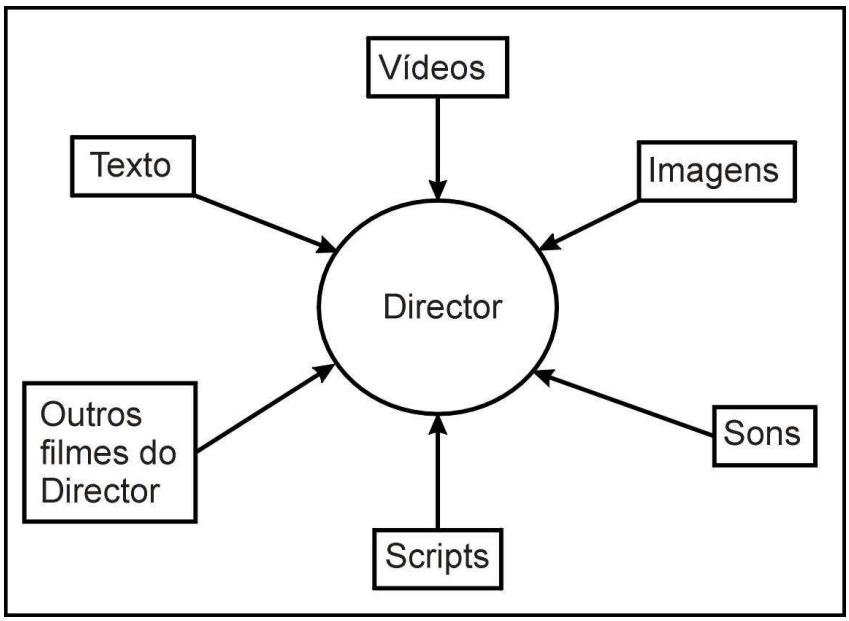
A ferramenta utilizada para o desenvolvimento do protótipo é o software de autoria Macromedia Director 8, o qual tem sido muito utilizado para o desenvolvimento de softwares multimídia/hipermídia.

Desde a sua criação, o Director vem incorporando novas e importantes características, principalmente em relação a integração com a Internet. Na versão 8 esta integração vem mais acentuada possibilitando o desenvolvimento de aplicativos multiusuários, embora este recurso já estivesse disponível na versão 7. No entanto, a diferença é a nova versão Multiuser Xtra, foco principal deste trabalho, que em relação ao anterior oferece maiores possibilidades de acompanhamento e controle das atividades realizadas pelo usuário.

4.1 APRESENTAÇÃO DO DIRECTOR

O Director 8.0 utiliza como metáfora o desenvolvimento de um filme ou peça de teatro. Dessa forma, o usuário atua como um “Diretor” do desenvolvimento da aplicação multimídia, selecionando as diversas mídias que irão fazer parte da aplicação final (Figura 1).

FIGURA 1 – ORGANIZAÇÃO DOS CAST MEMBERS PELO DIRECTOR

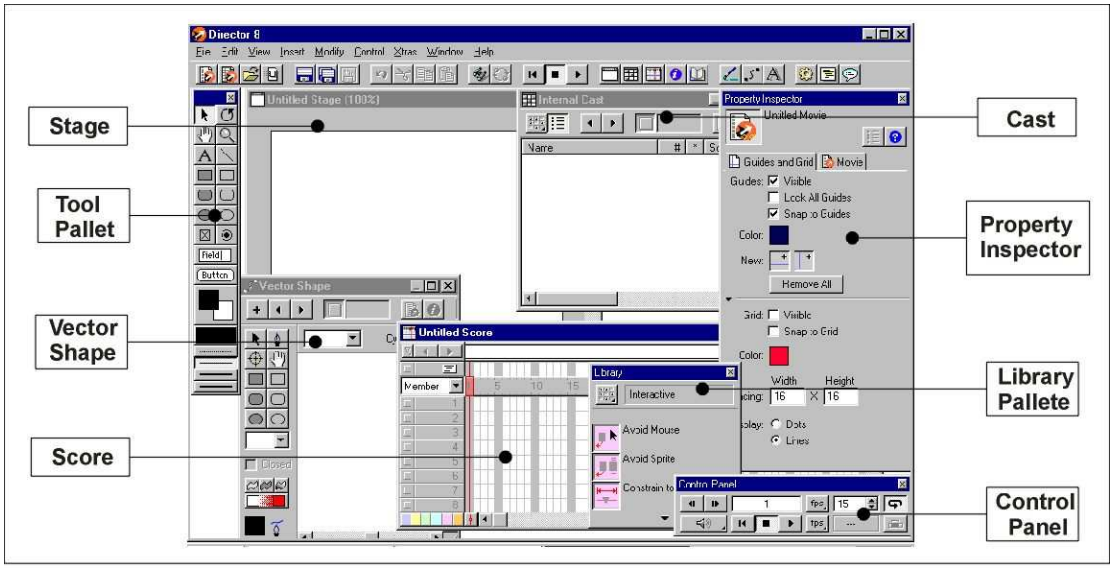


FONTE: Bizzotto, 2000

Em termos gerais, pode-se afirmar que o Director é constituído de duas partes interdependentes: o *Score* e a linguagem Lingo. O *Score* representa a parte que pode ser utilizada por usuários leigos para o desenvolvimento de aplicações simples. A linguagem Lingo é utilizada por usuários experientes e permite o desenvolvimento de aplicações complexas.

Independente do nível de experiência do usuário, ele irá interagir com alguns elementos básicos do Director 8.0, os quais podem ser visualizados na Figura 2 e descritos em maiores detalhes nos itens a seguir.

FIGURA 2 – JANELAS DO DIRECTOR 8



4.1.1 JANELA STAGE

A janela *Stage* é o palco onde o projeto é desenvolvido. Qualquer alteração feita através das janelas *Score*, ou *Cast*, irá se refletir no *Stage*. Na verdade, a metáfora do Director se materializa no *Stage*: tem-se um palco onde, através da utilização das ferramentas adequadas, o projeto existente no papel transforma-se em realidade. O Director 8.0 incorpora uma nova característica que é a possibilidade da alteração do “Grau de Zoom” no qual visualizamos o filme em desenvolvimento (Bizzotto, 2000).

4.1.2 JANELA CAST

Na janela *Cast (Internal Cast)* estão todos os atores (*Cast Members*) de um projeto e que correspondem ao elenco de uma peça teatral ou de um filme. Desta forma, qualquer imagem, som, vídeo, etc., que se incluir no projeto, irá aparecer na janela *Cast*. Esta janela possui uma característica muito importante que é a possibilidade de se usar cada *Cast Member* mais de uma vez em uma mesma cena, sendo que cada um deles pode ter comportamento e aparência diferentes. Isto poderia ser comparado a situação de um ator em um filme, contracenando com ele mesmo.

4.1.3 JANELA SCORE

A janela *Score (Untitled Score)* pode ser vista como o roteiro de um filme, onde os *Cast Members* são organizados de acordo com o planejado. Em termos gerais, o *Score* é uma linha de tempo, onde as cenas são organizadas de acordo com uma seqüência pré-definida. Esta seqüência pode ser tanto linear quanto não-linear. Esta janela é semelhante a uma planilha eletrônica e esta dividida em linhas, que são os canais, e por colunas, que são os *frames.* A intersecção de uma linha com uma coluna, denomina-se célula (Bizzotto, 2000).

Além dos canais usuais de gerenciamento dos *Cast Members*, possui outros seis canais: de ***tempo*** responsável pelo gerenciamento do tempo de apresentação e acontecimentos no *Stage;* de *Palette* que posibilita a utilização de um certo grupo de cores (*palette*) em dado filme; de ***transição***: responsável em gerar um efeito de transição de uma cena para outra; dois canais de ***som*** possibilitando a utilização de sons ou narrações e o canal de ***Script*** que permite a programação em Lingo (Hilgert, 2000).

4.1.4 JANELA PROPERTY INSPECTOR

Nesta janela se faz as alterações das propriedades relacionadas aos elementos em questão (*Sprite, Cast Member, Stage,* etc). Ela proporciona agilidade no desenvolvimento por ser “sensível ao contexto” (Bizzotto, 2000).

4.1.5 JANELA TOOL PALETTE

A janela *Tool Palette* permite a criação de ***Shapes,*** que são *Cast Meembers* que requerem pouca memória para serem utilizados, característica importante para a utilização em aplicações na Internet. A partir desta janela, podemos criar formas básicas (elipses, círculos, quadrados, etc) e botões. Com o uso desta janela, a criação do *Cast* Member é feita diretamente no *Stage* (Bizzotto, 2000).

4.1.6 JANELA CONTROL PANEL

Esta janela tem seu funcionamento semelhante ao controle de um vídeo cassete. que além dos comandos normais ainda 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).

4.1.7 JANELA VECTOR SHAPE

É através desta janela que podemos criar "Formas Vetoriais" (*Vector Shape*), que oferecem possibilidades de desenho (curvas, degrades, etc) muito superiores à *Tool Palette*. Porém, seu tamanho é um pouco maior que os *Shapes* (Bizzotto, 2000).

4.1.8 JANELA LIBRARY PALETTE

Esta janela contém os *Behaviors* (*scripst*) já prontos que se pode utilizar na construção de filmes, de forma a inserir interatividade, animação, efeitos, etc (Bizzotto, 2000).

4.2 O LINGO

Lingo é o nome da linguagem utilizada pelo Director, para a produção mais efetiva junto ao usuário. É uma linguagem de "*script*", cuja sintaxe e construção se aproxima da língua inglesa falada. Os *scripts* podem ser compostos de um único comando ou uma seqüência deles, dispostos em conjuntos, similares a parágrafos. A complexidade varia de acordo com a aplicação (Hilgert, 2000).

Como nas demais linguagens de programação, o Lingo segue algumas regras (Hilgert, 2000):

a) todo *script* é formado por *handlers* que são os equivalentes as sub-rotinas ou funções em outras linguagens. Um *handler* inicia com *on* e termina com *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 procedência de *handlers*:
   - *scripts* associados a eventos primários tais como:
     - *keydownscript;*
     - *mousedownscript;*
     - *mouseupscript;*
     - *timeoutscript;*
     - *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 a um filme (*movie script*);
e) orientação a objetos.

Segundo Small (1996), o Lingo não é uma linguagem orientada o objetos nativa, mas com o passar do tempo foi incorporando recursos que permitem a programação orientada a objetos. Através das variáveis globais privadas, cada objeto possui características próprias. A terminologia utilizada na orientação a objetos difere do Lingo, conforme demonstrado na Quadro 2:

QUADRO 2 – TERMOS LINGO EQUIVALENTES EM OO

Terminologia LINGO	**Equivalência OO**
Parent script	Classe
Child object	Instância de classe
Property variable	Variável de instância
Handler	Método
Ancestor script	Super classe

Fonte: adaptado de Hilgert(2000).

Características que possibilitam a orientação a objetos (Small, 1996):

a) a função "*New*" permite cópias idênticas de um determinado objeto através de um *script* denominado *Parente Script*;
b) através das propriedades, *Property*, os objetos mantêm suas próprias variáveis globais privadas;
c) os objetos podem dividir propriedades ou *handlers* através do *Ancestor*.

4.2.1 OS SCRIPTS

Os *scripts* de Lingo podem variar quanto a sua localização e tipo (Bizzotto, 1999):

a) *Behavior script* são *scripts* associados a *Sprites* (*Sprite behaviors*) ou a *frames* (*frames behaviors*).
b) *Cast members scripts*: associados 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 o todo o filme. Assim eles são mais gerais, não se limitando a um dado *Sprite* ou conjunto de *Sprite* ou *frames*.

4.2.2 EVENTOS

Alguns eventos são restritos ao tipo de *script* correspondente, sendo assim, executados somente apartir do *movie script* a que pertencem, conforme Quadro 3 (Bizzotto, 1999).

QUADRO 3 – EVENTOS E TIPOS DE SCRIPTS

Movie script	**Frame script**	**Cast member script e Sprite script**
On startMovie	On enterFrame	On keyDown
On StopMovie	On exitFrame	On keyUp
On timeOut	On keyDown	On mouseDown
On idle	On keyUp	On mouseUp
On enterFrame	On mouseDown	On mouseEnter

Fonte: adaptado de Bizzotto (2000).

QUADRO 3 – EVENTOS E TIPOS DE SCRIPTS (cont.)

Movie script	**Frame script**	**Cast member script e Sprite script**
On exitFrame	On mouseUp	On mouseLeave
On keyDown	On mouseEnter	On mouseWithin
On keyUp	On mouseLeave	On rightMouseDown
On mouseDown	On mouseWithin	On rightMouseUp
On mouseUp	On prepareFrame	-
On mouseEnter	On cuePassed	-
On mouseLeave	On rightMouseDown	-
On mouseWithin	On rightMouseUp	-
On prepareFrame	-	-
On cuePassed	-	-
On rightMouseDown	-	-
On rightMouseUp	-	-

Fonte: adaptado de Bizzotto (2000).

4.2.3 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 sendo portanto, necessárias as informações quanto ao tamanho dos elementos da lista. O primeiro elemento da lista é contado como “1”, diferentemente de algumas linguagens que consideram o primeiro elemento “0”.

4.3 FUNÇÃO MULTIUSUÁRIO

Uma aplicação multiusuário no Director utiliza dois componentes básicos: o Multriuser Server e o Multiuser Xtra. Esses dois componentes possibilitam a troca de informações entre filmes, através da Internet ou redes menores (Figura 3).

FIGURA 3 – APLICAÇÃO MULTIUSUÁRIO

Através do Multiuser Server e Multiuser Xtra pode-se:

- A criação de um *chat* que permitirá a conversação *on-line*.
- A criação de quadro para desenho em que os usuários de uma mesma sessão possam colaborar entre si *on-line*.
- Elaborar uma apresentação que possa ser assistida por todos ao mesmo tempo.
- Executar um jogo interativo com diversos participantes.

O Multiuser Xtra é um aplicativo instalado em todos os computadores que fazem parte de uma rede onde filmes do Director serão utilizados. Funciona como um protocolo de comunicação que verifica se as informações a serem transmitidas: não possuem erros, prepara estas informações para serem transmitidas e as envia ao servidor. Ao receber informações ou mensagens de outros filmes o Multiuser Xtra verifica novamente se esta não contém erro e disponibiliza então para ser usada no filme.

O Multiuser Xtra ou somente Xtra, pode trocar as mensagens entre filmes através de três modos básicos:

- Enviando as mensagens ao servidor e este encaminhando para o(s) filme(s) destinatário(s).
- Enviando as mensagens diretamente a outros filmes através de uma conexão do tipo *peer-to-peer*.
- Enviando as mensagens a um servidor de texto do tipo correio eletrônico, *Internet Relay*, ou *Chat Server*. Neste caso faz-se necessário conhecer os comandos básicos para envio de mensagens através deste tipo de servidor.

O Xtra que incorpora o Director 8 amplia a linguagem Lingo adicionando novos elementos e comandos scripts dos filmes. As mensagens que o Xtra envia aos filmes para que fim eles foram criados. As mensagens podem ser compostas por vários tipos de dados como: *strings*, inteiros, números de ponto flutuante, cores, datas, pontos, *recs* e listas.

Através do Xtra também é possível alterar um *Cast Member* através das propriedades *media* ou *picture of member*. A utilização dessas propriedades permite o uso de *Cast Member* para a criação de filmes com interação *on-line*. Por exemplo, um *chat* com desenho onde os usuários compartilham e interagem ao mesmo tempo.

O Multiuser Server é um aplicativo que está instalado em um computador que terá a função de servidor. Este servidor passa a ser o “elo” de comunicação entre os filmes recebendo mensagens que são enviadas pelos mesmos, através do Xtra, verifica para quais filmes e/ou usuários se destinam e as envia aos mesmos. Esta comunicação pode ser de um filme para outras instâncias suas, como por exemplo, um filme que é um jogo de xadrez onde os jogadores trocam mensagens sobre a movimentação das peças ou a comunicação entre filmes diferentes, como por exemplo, em uma sala virtual de jogos onde um jogador de xadrez, através de um *chat*, conversa com um jogador de damas.

O Multiuser Server além de administrar o envio de mensagens disponibiliza outras funções que permitem:

- O armazenamento e recuperação de informações como o nome do usuário (*users names*) ou os dados referentes a um usuário (*profile).*

- A criação de grupos de usuários deforma que fiquem organizados em modos lógicos, como por exemplo, times que disputam entre si um jogo coletivo qualquer.
- Obter informações referentes ao próprio servidor como também, informações referentes a outros filmes.

4.3.1 CRIANDO FILMES MULTIUSUÁRIO

A criação de uma aplicação multiusuário no Director precisa ter na sua implementação *scripts* que permitem a realização de uma conexão quer seja, com um servidor, ou diretamente com outros filmes, caso for uma conexão do tipo *peer-to-pee,* para possibilitar a troca de mensagens entre os filmes. Estas mensagens possuem todas as informações necessárias para o funcionamento destes filmes.

Os *scripts,* responsáveis pelo controle das conexões e envio de mensagens, são implementados utilizando a linguagem própria do Director, que é o Lingo. O próprio Director possui uma biblioteca que contém diversos *scripts* chamados de *behaviors,* que executam estas e outras tarefas. Isto resulta principalmente, quando o problema reside no fator tempo, a criação de filmes com maior agilidade, facilitando muito o processo de implementação.

4.3.1.1 CONEXÃO AO SERVIDOR

Para se implementar os *scripts* responsáveis pela conexão ao servidor e envio de mensagens, devem ser seguidos os seguintes passos:

1) É necessário que um computador da rede assuma o papel de servidor e seu endereço (*network adress*) deve ser conhecido.
2) Criar uma instância do Xtra.
3) Implementar um ou mais *callbacks* para fazer o tratamento das mensagens que são recebidas de outros filmes. Estes *callbacks* são ativados através do comando *setNetMessage.*
4) O último passo para a conexão ao servidor é efetivar a conexão através do comando *connectToNetServer.*

No Quadro 4 é apresentado um exemplo para criação da instância do Xtra (passo 2), onde a declaração (*new* (*xtra "multiuser"*)) cria a instância do Xtra e atribui a uma variável

global (*gMultiuserInstance*). Cada instância criada corresponde a uma única conexão. Para fazer mais de uma conexão ao mesmo tempo é necessário criar um número de instâncias igual ao número de conexões.

NOTA: Quando a conexão for encerrada, para a variável que recebeu a instância do Xtra, deverá ser atribuído o valor "zero".

QUADRO 4 – CRIAÇÃO DA INSTÂNCIA DE XTRA

```
gMultiuserInstance :

global gMultiuserInstance
gMultiuserInstance = new(xtra"Multiuser")
```

Após criada a instância do Xtra, declara-se os c*allbacks* para fazerem os tratamentos das mensagens enviadas e recebidas da rede (passo 3). Os c*allbacks* são *handlers* (sub-rotina) que são implementados para executar toda vez que o filme recebe uma mensagem da rede através de um servidor ou de outro filme nas conexões do tipo *peer-to-peer*. É necessário que os *callbacks* sejam criados antes de se estabelecer uma conexão.

Toda mensagem recebida é considerada um evento que irá ativar um *callback.* Assim para cada tipo de mensagem recebida deverá ser implementado um *callback* que fará o tratamento de acordo com o conteúdo de cada mensagem.

No Quadro 5 temos uma instrução utilizando o comando *setNetMessage* que ativa o *handler defaultMessageHandler*. Este *handler* se encontra em um *cast member* do tipo *script* denominado "*Connection Script*". O *defaultMessageHandler* é utilizado para tratar de forma genérica, qualquer tipo de mensagem recebida. O primeiro parâmetro especificado com o comando *setNetMassageHandler()* é um símbolo para o *handler* que está sendo declarado. Um parâmetro é criado adicionando um sinal de # (sustenido) no início do nome do *handler*. O segundo parâmetro é o nome do *script object* onde está declarado o *handler*. Este *script object* pode ser do tipo *Cast MemberScript,* o nome de uma variável que contém uma instância de *behavior,* um *script* pai, um simples valor do Lingo a ser passado para os *scripts* globais.

QUADRO 5 – HANDLER DE MENSAGEM

```
errCode=gMultiuserInstance.setNetMessageHandler(#defaultMessag
eHandler, script “Connection Script”)


O handler de mensagem se pareceria:


on defaultMessageHandler
  global gMultiuserInstance
  newMessage = gMultiuserInstance.getNetMessage()
  member("messageOutput").text = newMessage
  if newMessage.errorCode <> 0 then
    alert "Esta mensagem contém um erro."
  end if
end
```

Além do tratamento genérico das mensagens recebidas, é possível fazer um tratamento individualizado das mensagens usando *handlers* específicos que serão ativados dependendo do conteúdo do campo *subject,* do campo remetente, ou ainda ambos os conteúdos podem servir de parâmetros para acionar determinado *handler.* Estes parâmetros são adicionados no comando *setNetMassage.* Ao se optar passar estes dados para o *handler* como argumentos, deve-se colocar no final do comando *setNetMassage*, após os parâmetros da mensagem, o número “1” que irá setar o comando para este fim. Caso não se coloque nada ao final dos parâmetros, este valor, por *default*, será “0” e desabilita esta condição.

Passando o conteúdos das mensagens como argumentos, se evita a necessidade de se criar um tratamento só para pegar a mensagem da fila de mensagens recebidas para só depois verificar o seu conteúdo, como aconteceu no Quadro 5, onde foi preciso utilizar o comando *guetNetMassage* para buscar uma mensagem na fila. Esta fila é criada pelo Xtra onde as mensagens que chegam são colocadas e esperam para serem atendidas.

O Quadro 6 exemplifica esta situação onde o *handler #guestMessageHandler* é executado quando uma mensagem chega contendo no *subject "Chat Text"* e como remetente "*Guest Speaker"* e passa estes conteúdos como argumentos, já que o comando foi setado em "1", diretamente para o *Cast Member ("messageOutput"*).*text*, economizando tempo e linhas de código.

QUADRO 6 – TRATAMENTO DE MENSAGENS COM SUBJECT E REMETENTE

```
errCode = gMultiuserInstance.setNetMessageHandler(#guestMessageHandler, script "Connection Script", "Chat Text", "Guest Speaker", 1)

O handler de mensagem simplificado:

on guestMessageHandler me, message
  member("messageOutput").text = message
  if message.errorCode <> 0 then
    alert "Incoming message from Guest Speaker contained an error."
  end if
end

Se você quer incluir o parâmetro inteiro sem especificar um subject ou um remetente, use espaços vazios para o subject e remetente.
```

```
errCode                                                    =
gMultiuserInstance.setNetMessageHandler(#guestMessageHandler,
script"Connection Script", "", "", 1)
```

Após todos os *handlers* responsáveis pelo tratamento das mensagens terem sido declarados, é possível agora estabelecer uma conexão com o servidor.

O Xtra que funciona como um tipo de protocolo de comunicação fará o envio de uma mensagem ao servidor contendo as seguintes informações: o nome do usuário, sua senha, o nome do filme, o endereço ou o nome do servidor e a porta de comunicação do mesmo.

O exemplo mostrado no Quadro 7 materializa esta situação onde foram usados os seguintes dados: o filme a ser conectado é o "*Tech Chat*", o usuário é "*Bob*" e sua senha "*MySecret*", o nome do servidor de exemplo é "*chatserver.mycompany.com*" e a porta de comunicação, por *default*, é 1626.

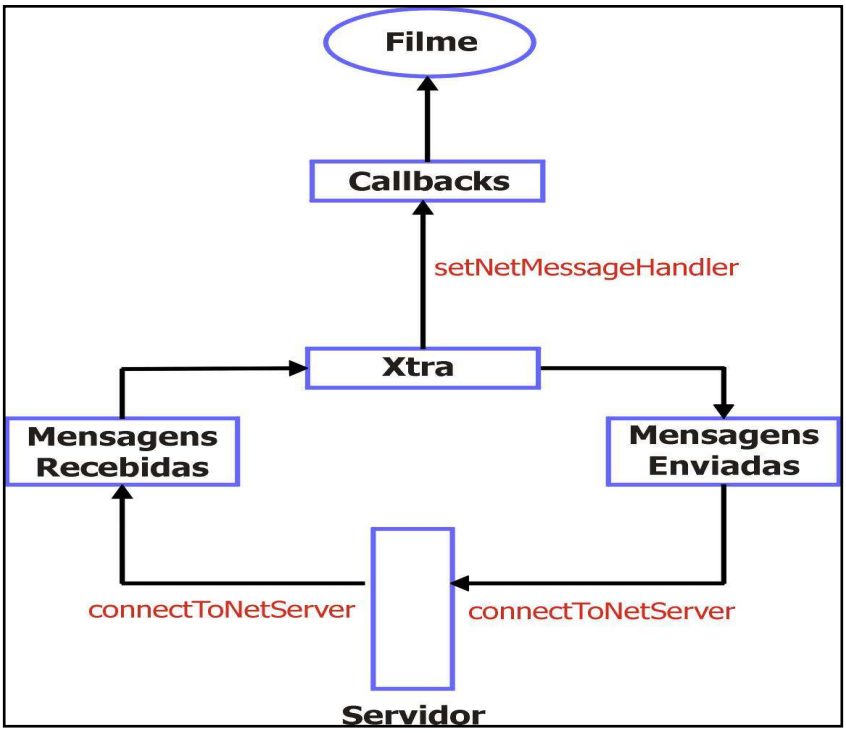
QUADRO 7 - XTRA CONECTANDO AO SERVIDOR

```
errCode                                                    =
gMultiuserInstance.connectToNetServer("Bob","MySecret", /
          "chatserver.mycompany.com", 1626, "Tech Chat")
```

O servidor, através do nome do filme indicado, irá verificar todas as instâncias do mesmo, ou seja, todos os outros usuários que estão utilizando o mesmo filme passarão a receber as mensagens referentes ao filme através do servidor.

Após a solicitação de conexão pelo usuário, o servidor irá checar as informações e se confirmada a conexão responderá ao usuário enviando uma mensagem contendo no campo *subject " connectToNetServer"*. Este *subject*, após recebido a mensagem pelo Xtra, ativará o *handler defaultMassageHandler* que fará o tratamento da mensagem. A Figura 4 mostra um esquema de uma solicitação de conexão ao servidor.

FIGURA 4 – PEDIDO DE CONEXÃO AO SERVIDOR



O comando *connectToNetServer()* possibilita também que o servidor verifique a localização de um filme que está conectado na Internet. Para isto ocorrer é necessário reescrever a instrução mostrada no Quadro 7 adicionando mais parâmetros.

O novo formato da instrução pode ser visto no Quadro 8 composta do nome do servidor e sua porta de comunicação seguida por uma lista de propriedades (nome do usuário, senha e nome do filme) e por final o novo parâmetro que é o modo de conexão. O modo de conexão é um parâmetro opcional dividido em dois tipos: o modo de conexão *#smus* e o modo de conexão *#text*.

QUADRO 8 – MODO DE CONEXÃO #SMUS.

```
errorCode = myConnection.connectToNetServer("chatserver.mycompany.com",/
            1626,[#userID:"Bob", #password:"MySecret", #movieID:"Tech/
            Chat"], #smus)
```

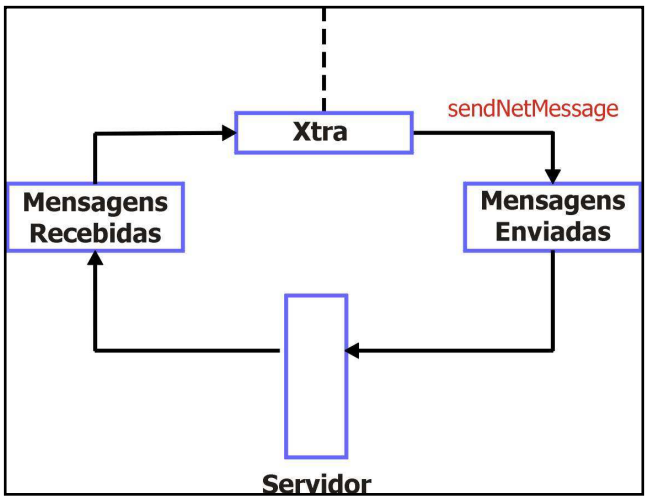
O modo de conexão *#smus* indica uma conexão normal a um servidor, enquanto o modo de conexão *#text*, indica uma conexão a um servidor que opera em modo texto e que possui tratamentos específicos.

Quando se utiliza o modo de conexão *#smus* no formato da instrução (Quadro 8), o Xtra, após a confirmação da conexão, irá enviar juntamente com o nome do usuário e a sua senha, o endereço da Internet onde o filme do usuário esta localizado. Este endereço é passado ao servidor no seguinte formato: *http//nomedoservidor.com/diretório/nomedofilme.dcr*. Isto permitirá ao servidor verificar se o filme se encontra realmente no local especificado pelo usuário proporcionando uma segurança adicional para os filmes distribuídos na rede.

4.3.1.2 ENVIANDO MENSAGENS

Estabelecida a conexão ao servidor com sucesso, é possível a partir de agora enviar mensagens a outros filmes conectados na rede. As mensagens a serem enviadas podem conter qualquer tipo de dado desde que aceitos pelo Lingo. Assim como as mensagens recebidas possuem o formato de lista de propriedades, pode-se optar em enviar as próprias mensagens também neste formato. A Figura 5 exemplifica o envio de mensagens através da função multiusuário do Director.

FIGURA 5 - ENVIANDO MENSAGENS



Para enviar os dados desejados através de uma mensagem é utilizado o comando *sendNetMessage*. Este comando permite enviar os dados de duas maneiras: através de uma

lista de propriedades ou enviar todos os parâmetros desejados separados entre si por vírgulas. Para exemplificar estas duas situações, ver Quadro 9 (Macromedia, 1999).

QUADRO 9 – USANDO O *sendNetMessage*.

```
errCode = gMultiuserInstance.sendNetMessage([#recipients:/
          whichUsersOrGroups,#subject:"Example Subject",#content:/
           whatMessage])
                                ou

errCode                                                    =
gMultiuserInstance.sendNetMessage("whichUsersOrGroups",/
            "Example Subject",whatMessage)
```

O comando *SendNetMassage* pode ser usado para enviar mensagens: a um usuário apenas do mesmo filme, a um grupo de usuários conectados a um mesmo filme, um usuário de um outro filme conectado ao servidor, ou verificar se um determinado usuário encontra-se conectado.

Para enviar uma mensagem a um usuário do mesmo filme, é declarada na instrução a instância do filme através de uma variável global criada para conter esta instância, o nome do usuário a quem se deseja enviar a mensagem, o *subject* da mensagem e por fim, a mensagem propriamente dita. No Quadro 10 é ilustrada esta instrução, onde um usuário envia uma mensagem a outro usuário do mesmo filme denominado de *"Bob"*, com o *subject "Chat Text"* e a mensagem *"Bem vindo ao nosso bate-papo"*.

QUADRO 10 – MENSAGEM AO USUÁRIO DO MESMO FILME

```
errCode  =  sendNetMessage(gMultiuserInstance,  "Bob",  "Chat
         Text",
           "Bem vindo ao nosso bate-papo.")
```

Desejando-se enviar uma mensagem a todos os membros de um grupo de usuários conectados no mesmo filme, é preciso que a instrução contenha o nome do grupo, um *subject* e a mensagem desejada. A declaração do nome do grupo deve ser antecedido pelo caracter

"@". O Quadro 11 apresenta um exemplo deste tipo de instrução tendo como nome do grupo "*MultimediaAuthors"*, *subject "Chat text"* e a mensagem "*O que vocês estão fazendo?"*.

QUADRO 11 – MENSAGEM PARA UM GRUPO

```
errCode                                                              =
gMultiuserInstance.sendNetMessage("@MultimediaAuthors",
         "Chat Text", "O que vocês estão fazendo?")
```

Quando a intenção é de se comunicar com um usuário de um outro filme conectado ao servidor, como em exemplo já citado anteriormente, em uma sala virtual de jogos, faz-se necessário a indicação do nome do filme para onde a mensagem deve ser enviada antecedido do caracter "@", que por sua vez é antecedido pelo nome do usuário do filme a quem se destina a mensagem, um *subject* e a própria mensagem. A seguir no Quadro 12 é apresentada esta instrução, onde, por exemplo, um usuário de um jogo de xadrez, em uma sala de jogos virtual, quer se comunicar com o jogador "*Chris"* que participa de um jogo de "D*amas"*. A mensagem terá como *subject "Pergunta"* e a mensagem será "*Quem ganhou?"*.

QUADRO 12 – USUÁRIO DE OUTRO FILME

```
errCode = gMultiuserInstance.sendNetMessage("Chris@Checkers",/
          "Pergunta", "Quem ganhou?")
```

Para se saber se um usuário se encontra conectado ao servidor, ou se ainda está conectado ao servidor, sem saber o filme em que ele compartilhava, a instrução seria implementada conforme é mostrado no Quadro 13, em que é mandada uma mensagem ao usuário *"Chris"*, contendo em seu *subject "Chamar"* e a mensagem "*Você está aí?"*, para verificar se este, se encontra conectado ao servidor. Neste caso o primeiro parâmetro passado da instrução é formado pelo nome do usuário que se deseja encontrar seguido de *@system.*

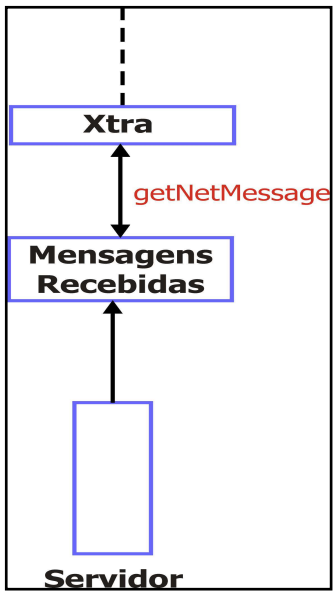
QUADRO 13 – LOCALIZANDO USUÁRIO

```
errCode = gMultiuserInstance.sendNetMessage("Chris@System", "Chamar",/
          "Você esta ai?")
```

4.3.1.3 RECEBENDO MENSAGENS

As mensagens que são enviadas do servidor para todos os filmes conectados, são armazenados em uma fila gerada pelo Xtra. Para buscar estas mensagens é usado o comando *getNetMassage* declarado no *handler* que fará o tratamento das mensagens (Figura 6).

FIGURA 6 – RECEBENDO AS MENSAGENS



A forma de declaração para o uso do comando *getNetMessage* é vista no Quadro 14, que traz na seqüência da instrução, os conteúdos que são atribuídos a variável *NewMessage* passados na forma de listas de propriedades.

QUADRO 14 – USANDO O *getNetMessage*.

```
      newMessage = gMultiuserInstance.getNetMessage

      Os conteúdos de newMassage serão uma lista de propriedades conforme
segue:


            #errorCode:0,#recipients:["Bob"],#senderID:"GuestSpe
            aker",#subject:"Chat/  Text",#content:"Hello.Welcome
            to our conversation.",/ #timeStamp:66114697]
```

Conforme já comentado, o tratamento das mensagens pode ser feitos de forma específica, através de um *handler* que só será ativado por um determinado tipo de mensagem fazendo o seu tratamento. Para que isto aconteça, o campo *#subject* ou as propriedades de *#sender*, deverão conter valores específicos que irão ativar o *handler* adequado ao tipo de mensagem enviada. No caso do parâmetro *#sender,* as ações a serem executadas pelo *handler* dependerão dos valores encontrados em cada uma das propriedades das mensagens recebidas.

Abaixo estão descritas as propriedades do parâmetro *#sender* definindo os seus conteúdos (Macromedia, 1999):

- #errorCode contém um valor do tipo inteiro indicando o código numérico de um determinado erro ocorrido. Através do comando g*etNetErrorString,* este código numérico é traduzido para uma mensagem textual exemplificando a falha.

- #recipients contém um valor do tipo *string* que indica o nome do usuário para quem a mensagem foi enviada.

- #senderID contém um valor do tipo *string* que indica o nome do usuário remetente da mensagem.

- #subject contém um valor do tipo *string* que indica o assunto da mensagem. Pode ser uma palavra apenas ou uma composição de palavras que resumem a mensagem de maneira que seja possível ativar um *handler* específico para o assunto descrito neste campo

- #content esta propriedade contem a mensagem propriamente dita, escrita por um usuário. Nesta propriedade a mensagem pode ser composta pelos seguintes tipos: *strings*, *rects*, listas, ponteiros, cores, datas e as propriedades *media of member* e *picture of member*.

- #timeStamp o conteúdo desta propriedade é formado por um valor do tipo inteiro que indica o tempo, em milissegundos, de funcionamento do servidor desde a sua inicialização. Pode ser utilizado para sincronizar eventos em várias instâncias de um filme ou determinar o tempo gasto pelo servidor para processar as mensagens.

Declarando o comando *getNetMessage ()* dentro de um *callback handler*, para "pegar" as mensagens da fila do Xtra, é possível implementar o *script* do *callback* com o objetivo de usar os conteúdos destas mensagens para fazer ações específicas, como por exemplo, usar os dados para definir a posição de um *sprite*, adicionar novas imagens no *stage* durante o filme, exibir um texto de *chat*, manter uma pontuação de um jogo que está sendo executado, etc.

O Multiuser Xtra também faz um controle das mensagens recebidas para verificar o tempo que estas permanecem inativas na fila de mensagens sem serem processadas. Isto pode gerar conflitos principalmente em filmes mais complexos onde o número de mensagens é elevado e o tempo inativo das mensagens aguardando para serem processadas, pode gerar problemas na execução do filme.

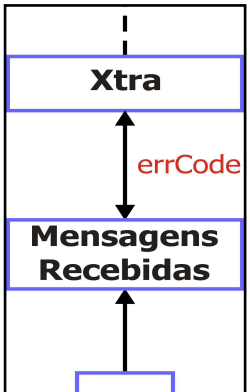
Através do comando *getNumberWaitingNetMessages*, obtem-se um valor que indica o número de mensagens recebidas que se encontram inativas na fila de mensagens recebidas no Xtra. Este valor pode ser usado na função *checkNetMessages()*, o qual verificará o número de mensagens em espera especificado, checando qual o *handler* determinado para o tratamento de uma mensagem e através do comando *setNetMessageHandler* enviará a mesma para o processamento no *handler* especificado.

4.3.1.4 VERIFICAÇÃO DE ERROS

Em filmes multiusuários em que os usuários se utilizam de uma rede para interagirem, o fluxo de informações que transita na rede é conseqüentemente alto, o que cria a necessidade de fazer a verificação da integridade das informações enviadas e recebidas.

Problemas como latência ou queda de conexões são comuns em sistemas de rede. Pode ocorrer quedas de energia deixando o servidor inoperante ou outros problemas de funcionamento de ordem física do servidor. A verificação de erros irá permitir detectar falhas e poder concertá-las a ponto do usuário final nem ser afetado pelo problema (Figura 7).

FIGURA 7 – VERIFICAÇÃO DE ERROS



SERVIDO

A maioria dos comandos utilizados pelo Multiuser Server e Multiuser Xtra são dependentes um do outro, o que torna ainda mais necessário e importante a verificação do bom funcionamento de cada comando, antes de se executar o próximo. Os resultados de execução de um comando podem ser síncronos ou assíncronos (Macromedia, 1999).

Um resultado gerado por um comando de forma síncrono, é quando a resposta do Multiuser Xtra é imediata, o que significa que o Xtra executou com sucesso a mensagem recebida e que os parâmetros fornecidos estavam corretos. Desta forma, é recebida uma mensagem contendo o código “0” (zero) que indica o sucesso da operação sem erros.

Um resultado gerado por um comando de forma assíncrono, é quando a resposta a uma mensagem só será fornecida após algumas operações terem sido executadas pelo servidor. Como no caso da solicitação de uma conexão através do comando *connectToNetServer,* em que o pedido é verificado pelo servidor e dependendo da situação, o servidor envia a resposta indicando se o pedido foi aceito ou recusado. O Quadro 15 exemplifica um resultado assíncrono em que a variável *errCode* recebe um valor do Xtra resultado do comando *connectToNetServer*. Caso haja parâmetros informados no comando *connectToNetServer* para o Xtra e esses não forem aceitos, o Xtra devolverá um resultado com a mensagem *nonzero* indicando a presença de erro e avisa com um alerta.

QUADRO 15 – RESULTADO ASSÍNCRONO

```
global gMultiuserInstance
errCode     =     gMultiuserInstance.connectToNetServer("Bob",
"MySecret",
```

```
"cha"tserver.mycompany.com", 1626, "Tech Chat")
if errCode <> 0 then
  alert "Problem with connectToNetServer" & RETURN & ¬
  gMultiuserInstance.getNetErrorString(errCode)
end if
```

Para conferir os resultados assíncronos retornados através do comando *#errorCode* sobre as propriedades de mensagens recebidas, é usado o mesmo *script* mostrado no Quadro 5.

Os erros podem ser gerados por instâncias de um filme onde uma destas instâncias pode se desconectar ou outros eventos podem interferir na comunicação eficaz do filme. Quando uma mensagem é enviada indicando a presença de um erro, ela traz essa informação na forma de um código numérico. Para converter este código numérico em uma mensagem descrita textualmente do problema ocorrido, usa-se o comando g*etNetErrorString*, conforme Quadro 16.

QUADRO 16 – CONVERTENDO O CÓDIGO DE ERRO EM TEXTO.

```
Global gmultiuserInstance
NewMessage = gMultiuserInstance.getNetMessage
ErrCode = newMessage.errorCode
Member(“errorText”).text = gMultiuserInstance.getNetErrorString(errCode)
```

4.3.1.5 OTIMIZANDO FILMES MULTIUSUÁRIO

O projeto de uma aplicação multiusuário deve assegurar que o filme responda às informações que chegam de outros filmes através da própria janela em que o filme está sendo executado. O filme deve enviar suas próprias mensagens assim que for necessário. As diretrizes seguintes ajudam a criar filmes mais ágeis e que respondam às mensagens enviadas de forma eficaz (Macromedia, 1999):

a) Minimizar a freqüência de mensagens – Projetar os filmes para enviar somente informações que são absolutamente necessárias. Para algumas aplicações, dependendo do conteúdo das mensagens, fará sentido juntar um grupo de pequenas mensagens e enviá-las todas juntas em uma mensagem maior.

b) Priorizar receber mais que enviar – Na maioria dos casos, deseja-se que os filmes recebam mensagens antes de enviar as suas próprias, pois os conteúdos das mensagens que chegam podem afetar a informação que se deseja enviar. Uma forma de realizar isto é aumentar a freqüência de exibição do filme. O Multiuser Xtra confere as mensagens que chegam durante os períodos inativos que ocorrem na troca de um *frame* para outro, assim sendo, quanto maior a freqüência de exibição, maior será a verificação do Xtra, o que ocasionará execuções mais rápidas das mensagens. Setando *idleHandlerPeriod* para 0 (zero), será maximizada a freqüência de eventos inativos permitindo ao Xtra conferir as mensagens durante e entre os *frames*. Usando a função *checkNetMassages()* pode-se coferir as mensagens a qualquer momento conforme desejado ou através do comando *getNumberWaitingNetMessages()* verificar quantas mensagens estão esperando na fila de mensagens do Xtra permitindo a especificação de um número de mensagens que se deseja processar através do parâmetro *checkNetMessage()*, assim é possível atender mais de uma mensagem enviada de uma só.

c) Minimizar atualizações do *Stage* – Geralmente, retirar S*prites* do S*tage* é muito mais demorado que enviar, receber ou processar mensagens. Em função disto, evita-se atualizar a tela toda vez que se recebe uma mensagem, por exemplo, em um jogo onde vários jogadores informam as suas posições, pode-se querer esperar para juntar todas estas informações primeiro para depois atualizar os *sprites* de uma só vez.

d) Enviar mensagens direcionadas – No caso de somente um usuário precisar de determinada informação de um filme com vários participantes, enviar somente uma mensagem a este usuário ao invés de enviar ao grupo inteiro.

e) Usar o *setNetMessageHandler()* – Colocando subjects diferentes para os diferentes tipos de mensagem que se está enviando, usa-se o *setNetMessageHandler()* para

processar as mensagens recebidas dependendo do subject, remetente ou ambos. O *Xtra message-dispatching* possui rotinas que executam mais rapidamente que o mesmo código escrito em Lingo.

4.3.1.6 A APLICAÇÃO SERVIDOR

Uma vez instalado o servidor, por default, o número de conexões permitidas simultaneamente são 50 podendo se alterar para 1000 através da edição do arquivo *Multiuser.cfg*. Na inicialização, o servidor vai ler o arquivo *Muliuser.cfg*, carregar o Xtra, e alocar suas conexões. Isto feito, o servidor estará apto a aceitar conexões e processar as mensagens dos filmes. Havendo inconsistência nos dados fornecidos no arquivo *Multiuser.cfg*, o mesmo ira encerrar a sessão.

Existem dois modos de visualização das atividades do servidor. O menu *View* que permite a ver mensagens na janela do servidor toda vez que certos eventos acontecem e o menu *Status* que permiti a visualização de informações sobre o número total de usuários conectados, filmes conectados, banco de dados utilizado, grupos criados, etc (Macromedia, 1999).

4.3.1.7 CONFIGURAÇÕES DO SERVIDOR

Muitos aspectos de como o servidor devera operar podem ser alterados editando o arquivo *Multiuser.cfg*. Características separadas para diferentes filmes podem ser especificadas através da criação de arquivos "*.cfg*" adicionais. Editando parâmetros em quaisquer destes arquivos "*.cfg*", é possível (Macromedia, 1999):

a) alterar o tamanho de memória disponível para o tratamento de mensagens;
b) limitar o número de usuários que podem se conectar ao servidor, ou para um filme específico;
c) especificar quais tipos de usuários podem se conectar ao servidor;
d) especificar quais filmes podem se conectar ao servidor;
e) setar os níveis de permissão dos usuários para executar cada comando do servidor.

4.3.1.8 ARQUIVO MULTIUSER.CFG

Podem ser adicionados comentários a qualquer arquivo de configuração precedendo cada linha de comentário com o sinal de sustenido (#), podendo ser utilizado também no começo de uma linha de parâmetro opcional para impedir a leitura deste pelo servidor. Para parâmetros que podem levar mais de um valor, colocar um espaço e uma contra-barra (\) depois do primeiro valor e colocar o próximo valor na próxima linha. Se o servidor não reconhecer um nome atribuído ou algum valor for nulo, a conexão será encerrada. Para que as mudanças no arquivo *Multiuser.cfg* tenham efeito, o servidor deverá ser reiniciado. Todos os itens possíveis de edição podem ser verificados em Macromedia (2001).

4.3.1.9 ARQUIVOS DE CONFIGURAÇÃO DE FILME

Cada filme diferente que se conecta ao servidor também pode ter o seu próprio arquivo de configuração. Os parâmetros podem ser os mesmos utilizados para o arquivo *Multiuser.cfg* do servidor. O arquivo de configuração do filme é lido pelo servidor quando a primeira instância do filme é conectada. Este arquivo deve receber o mesmo nome que o filme usa para se conectar ao servidor e colocar este no arquivo *Multiuser.cfg*. se o servidor não reconhecer o nome dado ou uma configuração é nula, os usuários não poderão se conectar com o filme (Macromedia, 1999).

5 METODOLOGIA E TÉCNICA DE MODELAGEM ORIENTADOS A OBJETOS

O desenvolvimento de software baseado em Orientação a Objetos, apresenta algumas vantagens em relação ao desenvolvimento convencional. Sendo as principais:

a) reusabilidade;
b) redução de erros;
c) redução de manutenção;
d) interações mais fáceis e mais rápidas.

Para entender a Orientação a Objetos é necessário antes entender alguns conceitos básicos desta metodologia, que são:

a) classe;
b) objeto;
c) método;
d) mensagens;
e) associação;
f) agregação;
g) herança;
h) encapsulamento.

5.1 CLASSE

As classes descrevem um conjunto de objetos (família) do mesmo tipo e com a mesma estrutura interna. Inclui informações que são os atributos, e comportamentos que são os métodos ou operações dos objetos. Os objetos são as estâncias da classe (Martin, 1995).

5.2 OBJETO

Segundo Martin (1995), um objeto pode ser qualquer coisa, real ou abstrata, a respeito da qual se armazenam dados e os métodos que os manipulam. Pode ser composto de outros objetos, que por sua vez podem ser compostos de outros objetos. O objeto preocupa-se tanto com seus dados quanto com os métodos com os quais os dados são manipulados.

5.3 MÉTODOS

Os métodos especificam a maneira pela qual os dados de um objeto são manipulados. Os métodos de um tipo de objeto referenciam somente as estruturas de dados desse tipo de objeto, não devendo acessar diretamente as estruturas de dados de outro tipo de objeto. Para acessar a estrutura de outro objeto deve ser enviada a mensagem a este objeto. Dessa forma, um objeto é qualquer coisa com suas propriedades representadas pelos tipos de dados e seu comportamento representado por métodos (Martin, 1995).

5.4 MENSAGENS

Para que um objeto realize alguma coisa, envia-se uma solicitação, que faz com que uma operação seja invocada. A operação executa o método apropriado e opcionalmente, retorna uma resposta. A mensagem que constitui a solicitação contém o nome do objeto, o nome da operação e, às vezes, um grupo de parâmetros.

5.5 ASSOCIAÇÃO

Relações que descrevem vínculos entre classes. É um relacionamento estrutural que especifica que objetos de uma classe estão conectados a objetos de outras classes. A associação pode ser rotulada (nomeada) e indica-se cardinalidade.

5.6 AGREGAÇÃO

Uma forma especial de associação entre o “todo” e suas “partes”. Por exemplo, um computador (o todo) é formado por monitor, teclado, mouse e CPU (as partes). Não se distingue um “todo” de uma “parte” e não impõe que a vida das “partes” esteja relacionado com a vida do “todo”. Os objetos “partes” não podem ser destruídos por qualquer objeto diferente do objeto de agregação que o criou.

5.7 HERANÇA

É quando uma classe genérica se divide em classes mais especializadas. Ou seja, a classe base, também chamada de superclasse ou pai, possui atributos, serviços e agregações

comuns e compartilhadas por classes mais especializadas, as subclasses ou filho, que adicionam ainda mais elementos Martin(1995).

5.8 ENCAPSULAMENTO

Para Martin (1995), encapsulamento "*é o ato de empacotar ao mesmo tempo dados e métodos*". O objeto esconde os seus dados de outros objetos e permite que os dados sejam acessados por intermédio de seus próprios métodos. O encapsulamento protege os dados contra adulterações e oculta os detalhes de sua implementação interna aos usuários de um objeto. Os usuários entendem quais operações do objeto podem ser solicitadas, mas não conhecem detalhes de como a operação é feita.

5.9 RELACIONAMENTOS

Os relacionamentos entre os objetos são demonstrados através dos símbolos mostrados na Quadro 17.

QUADRO 17 – SÍMBOLOS DOS RELACIONAMENTOS ENTRE OBJETOS.

———————————	Associação – relacionamento entre objetos.
∆	Herança – os objetos relacionados ao objeto acima, herdam as características.
◊	Agregação – o objeto relacionado a ele depende para existir.
1	Cardinalidade – o objeto se relaciona exatamente uma vez.
*	Cardinalidade – o objeto se relaciona várias vezes (zero ou mais).
0..1	Cardinalidade – a relacionamento do objeto é opcional (zero ou um).
m..n	Cardinalidade – valores especificados do relacionamento do objeto.

5.10 TÉCNICA DE MODELAGEM A OBJETOS (UML)

Nascido da união dos três principais métodos criados por James Rumbaugh, Grady Booch e Ivar Jacobson utilizados para orientação a objetos, a UML é uma linguagem de modelagem, assim denominada por não possuir um processo de desenvolvimento do software.

A UML é composta por vários diagramas, dos quais podemos citar (Furlan, 1998):

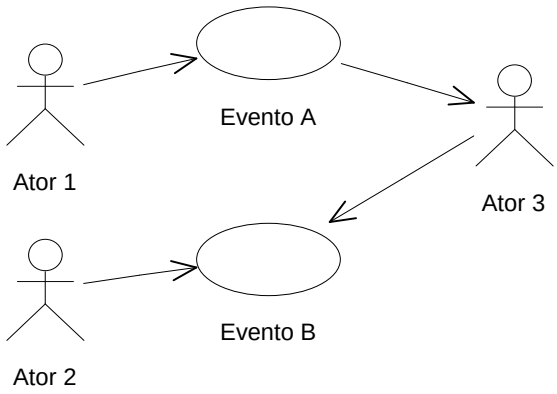
a) diagrama de caso de uso: mostra os limites de um sistema e suas principais funções;
b) diagrama de classe: representa a estrutura estática de um sistema;
c) diagrama de seqüência: dão ênfase à ordenação de mensagens;
d) diagrama de colaboração: serve para ilustrar a realização dos casos de uso;
e) diagrama de estado: modela o comportamento dos objetos;
f) diagrama de componente: apresenta a implementação física do sistema;
g) diagramas de implementação: apresenta a arquitetura do sistema;
h) esteriótipos: serve para criar extensões.

No referente trabalho, utilizou-se apenas os três primeiros diagramas, com o auxílio da ferramenta Case Rational Rose.

5.11 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso (Figura 8) é a especificação de seqüências de ações que um sistema, subsistema, ou classe podem realizar. Um caso de uso captura alguma função visível ao ator e, em especial, busca atingir uma meta do mesmo. Um ator interage com o sistema podendo este ser um usuário, dispositivo ou outro sistema. Podem incluir seqüências alternativas, ou seqüências excepcionais (de erro) (Furlan, 1998).

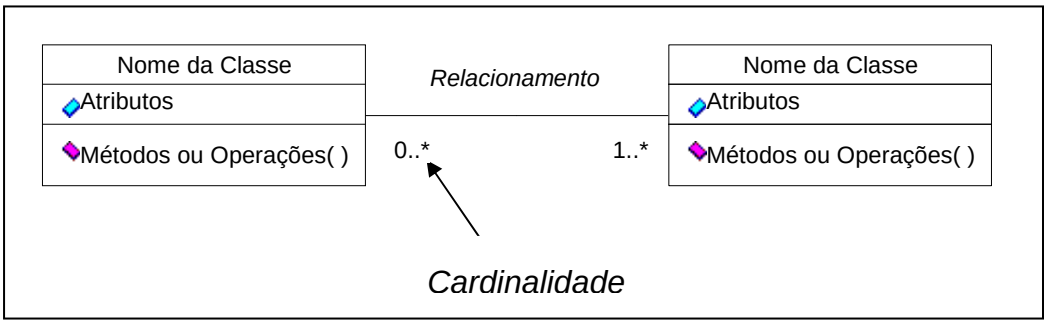
FIGURA 8 – DIAGRAMA DE CASO DE USO



5.12 DIAGRAMA DE CLASSE

O diagrama de classe mostra as classes, interfaces e o relacionamento entre os elementos. Denota a estrutura estática do sistema e as classes representam as entidades que são manipuladas por este sistema. O diagrama é considerado estático pois a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema (Furlan, 1998). O diagrama de classes pode ser visto na Figura 9.

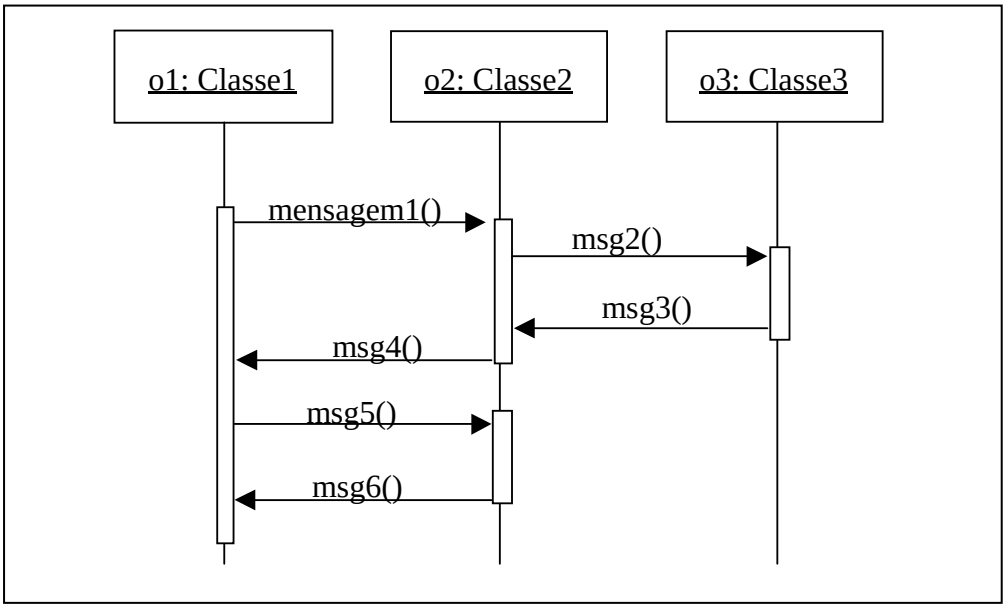
FIGURA 9 – DIAGRAMA DE CLASSES



5.13 DIAGRAMA DE SEQÜÊNCIA

Apresenta a interação de seqüência de tempo dos objetos que participam na interação. As duas dimensões de um diagrama de seqüência consistem na dimensão vertical (tempo) e na dimensão horizontal (objetos diferentes). O diagrama de seqüência mostra a colaboração dinâmica entre um número de objetos e o aspecto desse diagrama é mostrar a seqüência de mensagens enviadas entre objetos (Furlan, 1998). Na Figura 10 é mostrado este tipo de diagrama.

FIGURA 10 – DIAGRAMA DE SEQÜÊNCIA



6 DESENVOLVIMENTO DO PROTÓTIPO

Serão apresentados neste capítulo, a especificação do protótipo e a sua implementação.

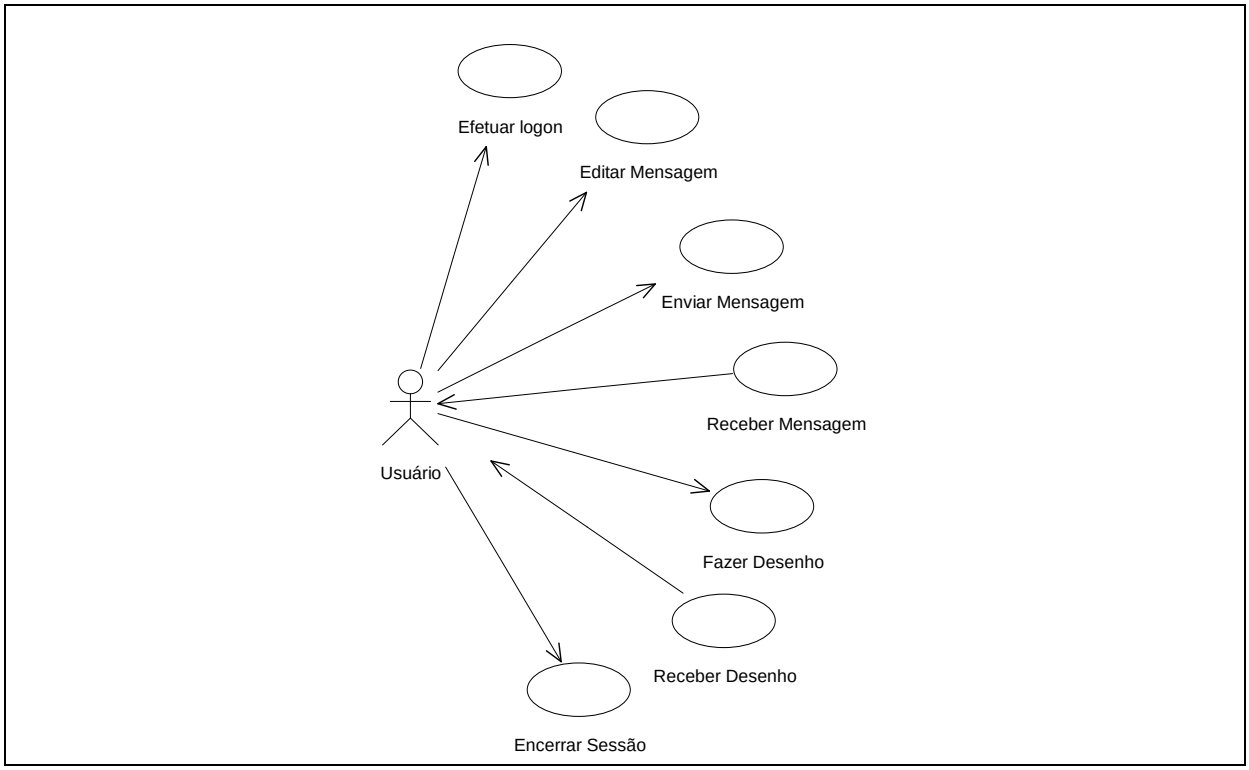
6.1 ESPECIFICAÇÃO DO PROTÓTIPO

Para a especificação do protótipo foram utilizados os diagramas: de Caso de Uso, diagrama de Classes e o diagrama de Seqüência, conforme o padrão UML e com o auxílio da ferramenta Rational Rose.

6.1.1 DIAGRAMA DE CASO DE USO DO PROTÓTIPO

Através deste diagrama (Figura 11) pode-se verificar a interação do usuário com o sistema de *chat* através de uma seqüência de ações.

FIGURA 11 – DIAGRAMA DE CASO DE USO DO PROTÓTIPO



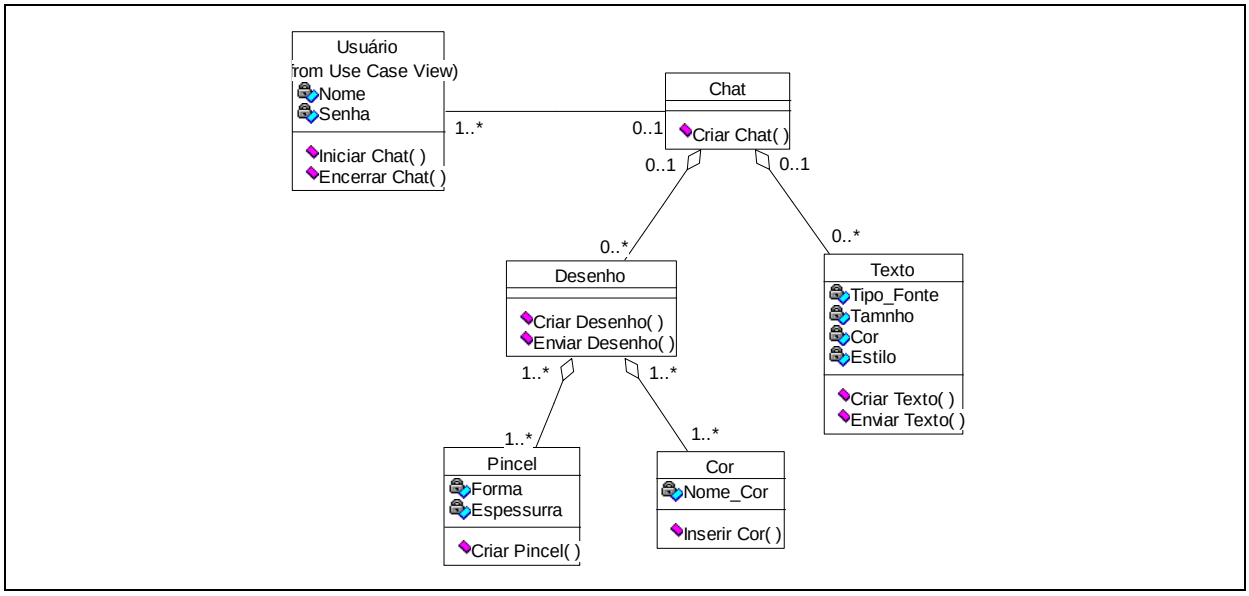
Ao ser utilizado o sistema pelo usuário, são gerados os seguintes eventos:

1. Efetuar logon: fornecendo o seu nome e senha, o usuário solicita uma conexão para entrada no *Chat*.
2. Editar mensagens: o usuário digita a mensagem a ser enviada.
3. Enviar mensagem: o usuário envia a mensagem digitada a todos os participantes conectados na sessão de *chat*.
4. Receber mensagem: o usuário visualiza as mensagens enviadas durante a sessão de *chat*.
5. Fazer desenho: o usuário seleciona o pincel (grosso ou fino) e a cor e executa o desenho e no término o desenho é enviado.
6. Receber desenho: no próprio quadro de desenho, o usuário visualiza o desenho compartilhado.
7. Encerrar a sessão: o usuário sai do sistema.

6.1.2 DIAGRAMA DE CLASSES DO PROTÓTIPO

A Figura 12 mostra a estrutura do protótipo através de suas classes, interfaces e relacionamentos entre os objetos do sistema.

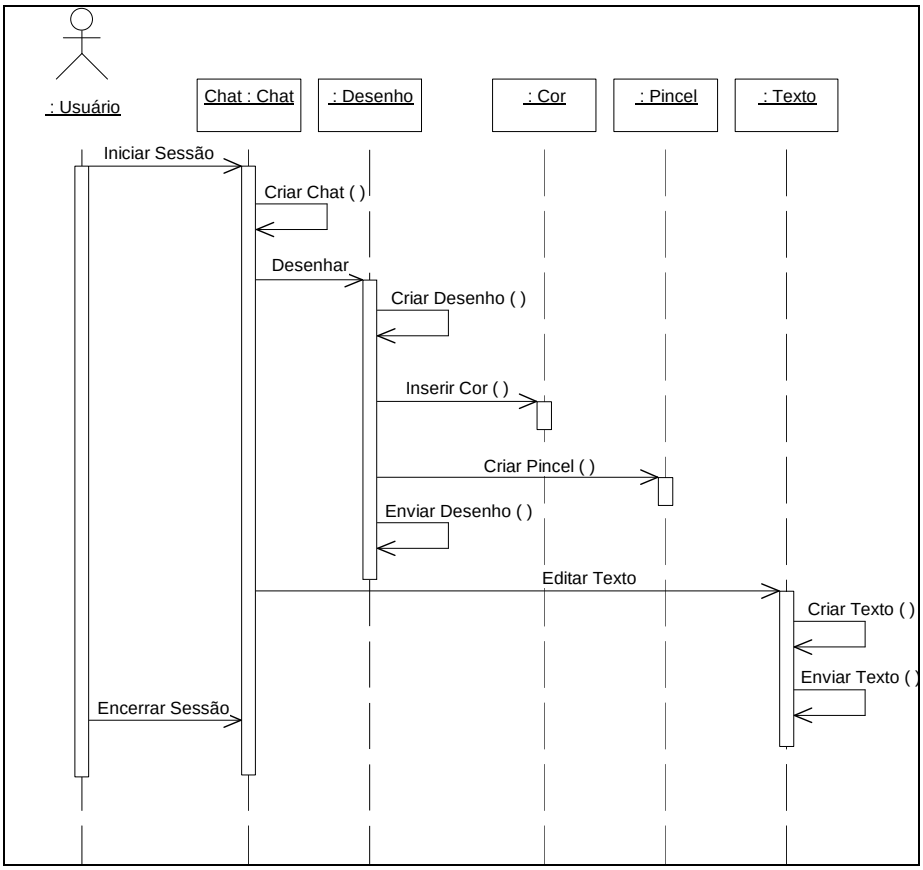
FIGURA 12 – DIAGRAMA DE CLASSES DO PROTÓTIPO.



6.1.3 DIAGRAMA DE SEQÜÊNCIA DO PROTÓTIPO

Através do diagrama de seqüência é possível observar as interações entre os objetos durante uma sessão de *chat* com desenho, mostrado na Figura 13.

FIGURA 13 – DIAGRAMA DE SEQÜÊNCIA DO PROTÓTIPO.



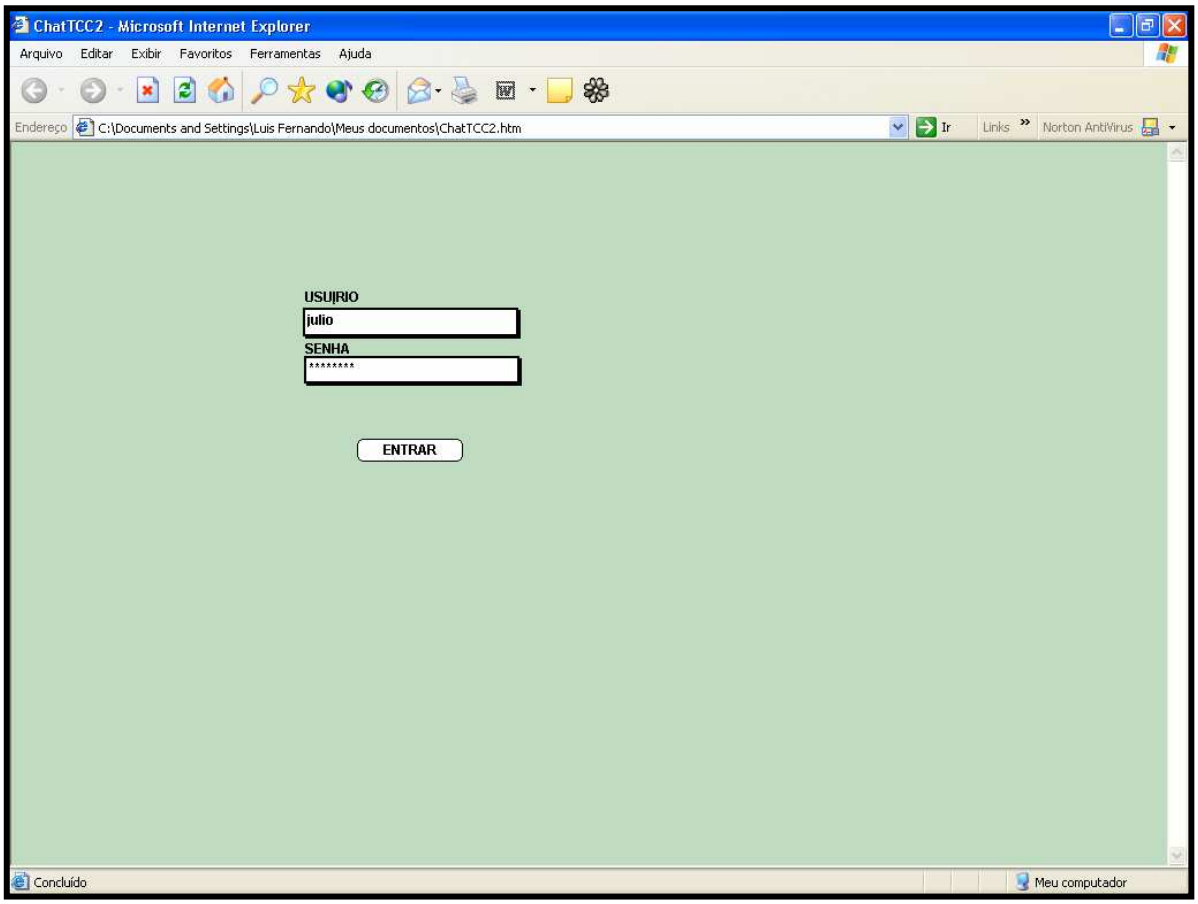
6.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste tópico serão apresentados as telas, comentários e os códigos implementados mais relevantes.

6.2.1 INICIANDO A CONEXÃO

Ao iniciar o filme da aplicação *chat*, será aberta a janela de navegação do *browser* padrão utilizado na máquina do usuário, onde aparecerá a tela de conexão ao servidor solicitando ao usuário que informe o seu nome e a sua senha (Figura 14). O pedido de conexão se confirma com o usuário clicando no botão “ENTRAR”.

FIGURA 14 – TELA DE CONEXÃO



Inicialmente, ao usuário abrir o filme de *chat*, são inicializadas todas as variáveis e ativados os comandos necessários para se estabelecer a conexão e fazer os tratamentos das

mensagens geradas pelo filme, através da função *F_Inicializar*, como é mostrado no Quadro 18.

QUADRO 18 – HANDLER DE INICIALIZAÇÃO

```
on F_Inicializar
   -- Inicialização de variáveis de sistema
  G_movie          =  "Chat"
  G_servidor       =  "127.0.0.1"
  G_sala           =  "@AllUsers"
  G_usuario        =  "gandhi"
  G_senha          =  "123456"
  G_porta          =  1626
  G_timer          =  []
  G_lista          =  []
  G_numlista       =  []
  G_grupos         =  []
  G_gruposusuario  =  []
  G_erro           =  0
  G_scrollchat     =  14

-- Inicialização dos timers
  Repeat with L_numero = 1 to 100
    add G_timer,0
  end repeat

-- Definição de handlers de scripts e conexão
  H_conexao = 0
  H_conexão = new(xtra “Multiuser”)

-- Mensagems gerais executam o “defaultmessagehandler”
  G_erro = H_conexao.SetNetMessageHandler(#defaultMessageHandler,script
“Scripts Básicos”)
  F_VerificaErro

-- Mensagem do Chat executa o “Chathandler”
  G_erro = H_conexao.SetNetMessageHandler(#ChatHandler,script "Scripts
Básicos","TextoChat","",1)
  F_VerificaErro

end F_Inicializar
```

Após inicializado o sistema, o usuário efetua o pedido de conexão que será realizado pela função *F_Connect*, mostrada no Quadro 19.

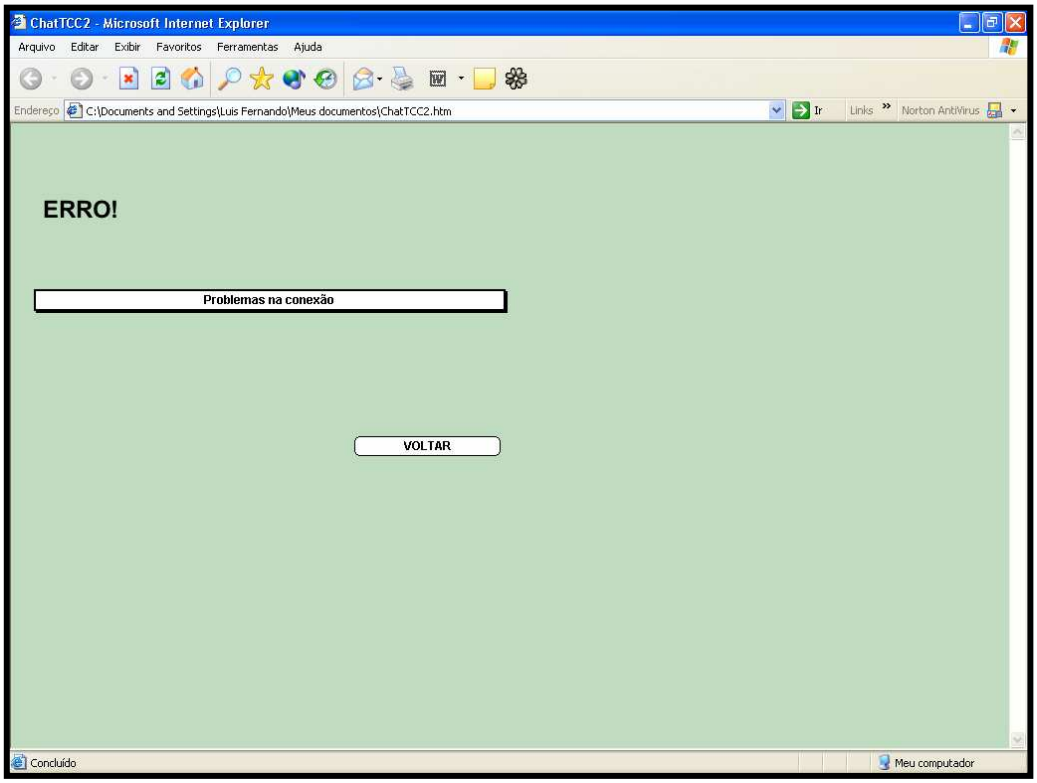
QUADRO 19 – PEDIDO DE CONEXÃO

```
on F_Connect
  -- Tenta conectar com o servidor
  G_erro = H_conexao.ConnectToNetServer(G_usuario,G_senha,G_servidor,/
           G_porta,G_movie)
  F_VerificaErro
end F_Connect
```

6.2.2 TRAMENTO DE ERROS

Caso haja algum problema com a conexão, surgirá um aviso de erro em uma nova tela criada para esse fim, indicando qual foi o problema ocorrido. O usuário deverá clicar no botão “VOLTAR” para retornar a tela inicial de conexão. A tela de mensagens de erro será aberta sempre que alguma anormalidade ocorrer em relação execução do filme (Figura 15).

FIGURA 15 – TELA DE MENSAGENS DE ERRO.



O código responsável pelo tratamento deerros do protótipo é apresentado no Quadro 20.

QUADRO 20 – ROTINA TRATADORA DE ERROS

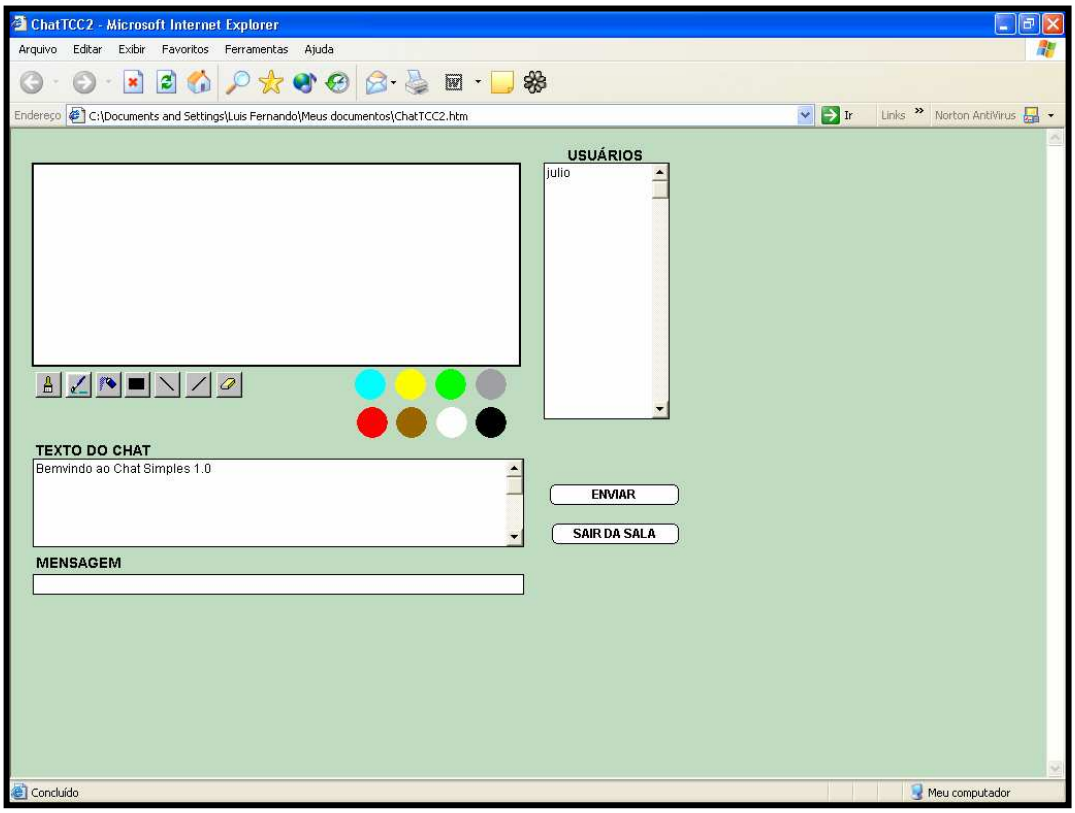
```
on F_VerificaErro
  if G_erro<>0 then
    case G_erro of
      -2147216221:
        G_MODO = "ERRONICK"
        G_textoerro = "Nick inválido ou existente"
      -2147216220:
        G_MODO = "ERROSENHA"
        G_textoerro = "Senha inválida"
      -2147216217:
        G_MODO = "ERROLOTADO"
        G_textoerro = "Servidor Lotado"
      -2147216214:
        G_MODO = "ERROCONEXAO"
        G_textoerro = "Conexão Inexistente"
      -2147216216:
      otherwise
        G_MODO = "ERROGERAL"
        G_textoerro = "Problemas na conexão"
     end case
    -- executa a rotina de tratamento
    F_TratamentoErro
  end if
end F_VerificaErro

on F_TratamentoErro
  -- redirecionado para um frame com o label "erro"
  put G_MODO,G_erro
  go to "ERRO"
end
```

6.2.3 SESSÃO DE CHAT

Se a conexão foi efetuada com sucesso, será aberta a tela do *chat* propriamente dito. A partir deste momento o usuário pode optar em enviar as mensagens por texto, desenho ou ambos. Para enviar uma mensagem de texto, o usuário deve digitar a mesma no campo “MENSAGEM” e clicar no botão “ENVIAR” para transmiti-la a todos os usuários conectados no filme. Ao enviar a mensagem, no campo “TEXTO DO CHAT”, irá aparecer o texto que o usuário acabou de enviar, assim como, as mensagens enviadas pelos outros usuários que participam do *chat* (Figura 16).

FIGURA 16 – TELA DE CHAT E DESENHO



Os Quadros 21 e 22 mostram os códigos para envio de mensagens e visualização das mensagens no campo “TEXTO DO CHAT” respectivamente.

QUADRO 21 – ENVIO DE MENSAGEM

```
global G_sala,H_conexao
on mouseUp me
  -- envia mensagem para a sala , desde que não esteja vazia
  L_texto=member("entrada").text
  if L_texto<>"" then
    F_EnviaChat(G_sala,L_texto)
    member("entrada").text=""
  end if
 end
```

QUADRO 22 – MENSAGENS ENVIADAS

```
on F_EnviaChat L_Sala,L_mensagem
  G_erro=H_conexao.SendNetMessage(L_Sala,"TextoChat",L_mensagem)
  F_VerificaErro
end F_EnviaChat


on ChatHandler me ,mensagem
  -- Todo o código abaixo controla o scroll automático da janela de chat
  L_texto = member("texto_chat").text
  member("texto_chat").text = L_texto & return & mensagem.senderID & " > "
& mensagem.content
  L_linhas = Member("texto_chat").linecount
  -- mantém um limite de linhas, apaga x de cada vez
  if L_linhas > 150 then
    member("texto_chat").text= line 75 to  150 of field "texto_chat"
    -- atualiza contagem de linhas para o scroll funcionar
    L_linhas = Member("texto_chat").linecount
  end if
  -- faz o scroll da tela automaticamente
  member("texto_chat").scrolltop =
member("texto_chat").linepostolocv(L_linhas-G_scrollchat)
end
```

Se o usuário desejar ilustrar sua mensagem com um desenho, ele escolhe um tipo de pincel (grosso ou fino) na barra de ferramentas, seleciona a cor desejada na paleta de cores e faz o seu desenho no quadro destinado a esse fim, mantendo o botão do mouse clicado enquanto executa o desenho. Ao soltar o botão do mouse, o desenho é enviado a todos os participantes que poderão interagir no mesmo desenho. Se o usuário desejar fazer alguma alteração no desenho, pode utilizar a borracha também disponível na barra de ferramentas. O Quadro 23 contém o código que realiza o envio do desenho. Este código faz parte do *behavior* CANVAS utilizado e que é responsável por todo tratamento da área do desenho.

QUADRO 23 – ENVIO DO DESENHO

```
on setTheMovieServerConnection(me, connectionInstance, ID) -
  -- enviando para todos os sprite pelo behavior Connect to Server somente as
  -- conexoes bem sucedidas
  -- * inicializa o behavior, se o ID for o mesmo do connectionID

  if ID = connectionID then
    if ilk(connectionInstance) = #instance then
      me.initialize(connectionInstance)
    else
      me.cleanUp()
    end if
  end if
end setTheMovieServerConnection
```

QUADRO 23 – ENVIO DO DESENHO (cont.).

```
on cleanup(me)
  errorCode = myConnection.setNetMessageHandler(0, me, canvasMessage )
```

```
  myConnection = void
end cleanup

on whiteboard_broadcast(me, dataList)
 --   enviando o behavior Canvas no mesmo sprite
  -- * envia uma mensagem com  o conteúdo <dataList> para o
  --   canvasGroup
  --
  -- <dataList> tem o formato:
  -- [#command: #symbol, #data: [<property list>]]
  ------------------------------------------------------------------------
    if ilk(myConnection) <> #instance then
    errorCode = me.initialize()
    if symbolP(errorCode) then
            return
    end if
  end if
    -- tenta enviar a mensagem de comando
  errorCode = myConnection.sendNetMessage( \
canvasGroup, \
canvasMessage, \
dataList \
)
  if errorCode then
    -- a mensagem não foi enviada, avise o autor
    explanation = myConnection.getNetErrorString(errorCode)
    call(#errorAlert, [me], #sendNetMessage, explanation)
    return errorCode
  end if
end whiteboard_broadcast
```

Ao entrar na tela de *chat*, o usuário é incluído no campo “USUÁRIOS” que mostra todos os participantes da sessão de *chat* conectados. Para encerrar a sua participação, o usuário clica no botão “SAIR DA SALA” e a aplicação é finalizada. O Quadro 24 apresenta o código que controla e mantém a lista de usuários.

QUADRO 24 – LISTA DE USUÁRIOS LOGADOS

```
on F_GetUsers L_Sala
 G_erro=H_conexao.SendNetMessage(“system.group.GetUsers”,”GetUsers”,L_Sala)
  F_VerificaErro
end F_GetUsers

on F_GetUserCount L_Sala
G_erro=H_conexao.SendNetMessage(“system.group.GetUserCount”,”GetUserCount”,
L_sala)
  F_VerificaErro
end F_GetUserCount
```

QUADRO 24 – LISTA DE USUÁRIOS LOGADOS (Cont.)

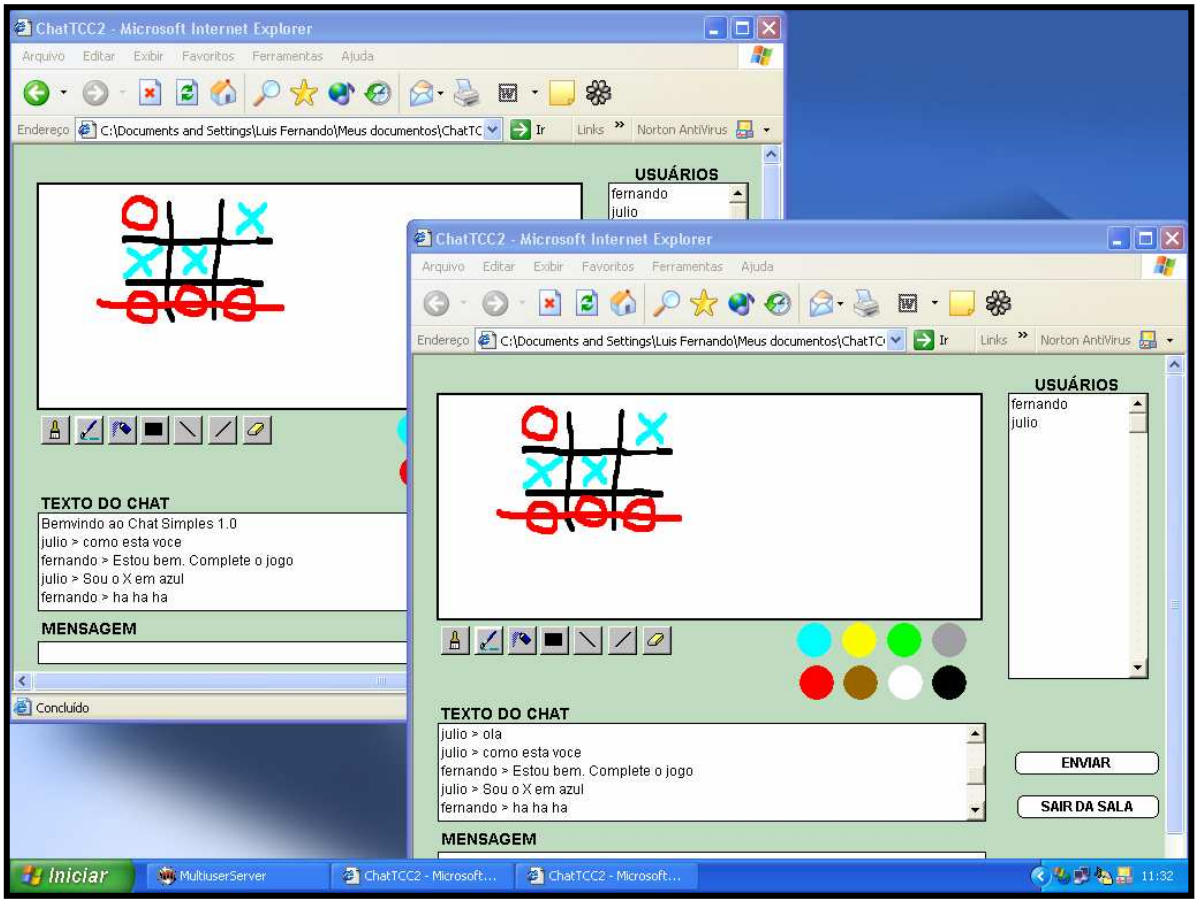
```
on F_AtualizarUsuarios L_sala
  G_erro=H_□onexão.SendNetMessage(L_sala,”AtualizaUsuarios”,L_sala)
  F_VerificaErro
end F_AtualizaUsuarios
```

```
on F_Lista
  L_total= G_lista.count
  texto=""
  repeat with t = 1 to L_total
    texto=texto&G_lista[t]&Return
  end repeat
  member("lista").text=texto
end F_Lista
```

6.2.4 USANDO O CHAT

A Figura 17 mostra a simulação da realização de uma sessão de *chat* realizada entre dois usuários que se conectaram ao filme e se comunicaram através do texto e usaram a tela de desenho para realizar o jogo da velha de forma interativa e *on-line*.

FIGURA 17 – USANDO O CHAT



7 CONCLUSÃO

O presente trabalho atingiu o seu objetivo principal com o estudo da função multiusuário do ambiente de autoria Macromedia Director 8.0, através da implementação de um protótipo que permitiu o compartilhamento, em tempo real, de um *chat* com mensagens em texto e também com a disposição de uma tela para desenho.

A função multiusuário do Director 8.0 é um assunto bastante complexo que exige um tempo maior que o que foi disposto para a realização deste trabalho, para que se possa conhecer a fundo tudo o que esta função pode permitir em relação a aplicações multiusuário.

Com o protótipo desenvolvido apresentado no Capítulo 6, foi possível criar uma aplicação multiusuário na forma de um *chat* com mensagens que podiam ser editadas na forma de texto e/ou escolher em compartilhar um desenho onde todos que participavam da sessão podiam interagir com a imagem gerada de forma *on-line*. Com os diferentes pincéis pode-se escolher a largura e formato da linha que se desejava desenhar, como também se optava por uma cor disposta na paleta de cores. Itens estes que enriqueceram a aplicação em termos de opções para a elaboração do desenho.

Com a implementação da lista de usuários logados, a sessão era fácilmente acompanhada sabendo-se sempre de todos os elementos que compartilhavam o *chat* naquele momento, sendo atualizada toda vez que alguém entrava ou saia do *chat*. A aplicação era aberta sempre no *browser* padrão da máquina se asemelhando a uma página da Internet, o que tornava o seu uso mais amigável.

Uma vantagem verificada, é que a Macromedia disponibiliza um servidor com capacidade de 50 conexões simultâneas, para testes dos softwares gerados por suas ferramentas que pode ser encontrado no site da própria Macromédia.

7.1 SUGESTÕES PARA TRABALHOS FUTUROS

A ferramenta se mostra bastante útil para as aplicações multiusuário o que leva a um campo muito grande de aplicações que ainda podem ser exploradas como:

- Na área de Química, por exemplo, podem ser gerados filmes que disponibilizem uma interação entre os alunos que podem colaborar no desenvolvimento e estudo de substâncias através das combinações dos elementos e suas ligações.
- Em laboratórios de Engenharia Elétrica, filmes que simulam as ligações em circuitos elétricos ou eletrônicos para análise do seu funcionamento pelos alunos. Ou em cursos de Engenharia Mecânica, onde os alunos possam simular a montagem e funcionamento de máquinas e desenvolvimento ferramentas.
- Além de desenhos, pode ser explorada a inclusão de fotos e/ou imagens e estas poderem ser alteradas pelos participantes do *chat*.

REFERÊNCIAS BIBLIOGRÁFICAS

BARKER, P. ***Computer – based training: an institutional approach*.** [s.l.] Education & Computing, 1992.

BIZZOTTO, Carlos Eduardo N. **Concepção, desenvolvimento e avaliação de um ambiente extensível para aprendizado cooperativo distribuído.** 1999. 134 f. Projeto (Qualificação ao Doutorado em Engenharia de Produção) – Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

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

CHAIBEN, Hamilton. **Hipermídia na educação,** Curitiba, nov. 1999. Disponível em:

<http:\ \www.cce.ufpr.br/~hamilton/hed/hed.htm>. Acesso em: 28 out. 2001.

FURLAN, José David. **Modelagem de objetos através da UML –** the unified modeling language**.** São Paulo: Makron Books, 1998. 329p.

GEORG, Cláudia. **Protótipo de software para aprendizagem de arranjos.** 2000. 85 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GRINKRAUT, Melaine Lerner. **Computador:** uma ferramenta a mais à disposição do professor. Informativo Sociedade de Psicologia de São Paulo, São Paulo: Ano 1, v.I, n.1,p.3, set.1996.

HEIDE, Ann; STILBORNE, Linda. **Guia do professor para Internet: completo e fácil.** 2.ed. Tradução Edson Furmankrewz. Porto Alegre: Artes Médicas Sul, 2000. 337p.

HILGERT, Celso João. **Protótipo de software para auxiliar ao aprendizado de cores e formas geométricas.** 2000. 87f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MACROMEDIA. ***Using the Shockwave Multiuser Server version 2.1 and Xtras,*** San Francisco, set. 1999. Disponível em: <http://www.macromedia.com/support/director/>. Acesso em: 25 ago. 2001.

MARTIN, James; ODELL, James J. **Análise e projeto orientado a objeto.** Tradução de José Carlos Barbosa dos Santos; revisão técnica de Ronald Stevis Cassiolato. São Paulo: Makron Books, 1995.

MEYER, Marilyn; BABER, Roberta; PFAFFENBERGER, Bryan. **Nosso futuro e o computador.** 3.ed. Tradução Edson Furmankiewicz. Porto Alegre: Bookman, 2000. 599p.

MOHANDAS, CK. **Interatividade: a real evolução da cultura,** [s.l.], mar. 2001. Disponível em: <http://www.superdownloads.com.br/matérias/20010319,53,1.html/>. Acesso em: 11 nov. 2001.

MUNDO DIGITAL. **A história da Internet,** São Paulo, jan. 1997. Disponível em: <http://www.uol.com.br/mundodigital/beaba/fase1.htm>. Acesso em: 22 set. 2001.

SANTOS, Andréa Carla Krieck dos. **Protótipo de um software de autoria para elaboração e aplicação de avaliações por computador.** 1997. 72 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SILVA, Elaine Quintino da. et al. **Computadores no ensino – uma abordagem voltada para o suporte aos professores no desenvolvimento de atividades didáticas.** In: Simpósio Brasileiro de Informática na Educação, 2000, Maceió. **Anais: As novas tecnologias da informação e comunicação na aprendizagem.** Universidade Federal de Alagoas, 2000. p.81-88.

SMALL, Peter. ***Lingo Sorcery: the magic of lists, objects and intelligent agents.*** London: Jonh Wiley and sons, 1996.

SZETO, Gong et al. **Interatividade na Web**. Tradução Marcos Vieira. São Paulo: Berkeley, 1997. 494p.

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

TAYLOR, R. P. ***The computer in the school: tutor, tool, tutee*.** Teachers College Press, New York, 1980.

VALENTE, J. Armando. **Análise de diferentes tipos de softwares usados na educação** – NIED – UNICAMP – In: III Encontro Nacional do PROINFO – MEC, 1998, Pirenópolis – GO.

VALENTE, José Armando. **O computador na sociedade do conhecimento.** Campinas: Ed. Do Núcleo de Informática Aplicada a Educação da Universidade Estadual de Campinas, 1999.

VALENTE, José Armando. **Diferentes usos do computador na educação.** [s.l.], mar. 1999. Disponível em: http://www.proinfo.br. Acesso em: 25 ago. 2001.

WILHELM, Pedro Paulo Hugo. **Uma nova perspectiva de aproveitamento dos jogos de empresa.** 1997. 136 f. Tese (Doutorado em Engenharia de Produção e Sistemas) – Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.