

# **Agente Gerenciador de Cursos a Distância via Internet**

Elaine Quintino da Silva

## **Orientador**

Prof. Dr. Dilvan de Abreu Moreira

*Dissertação<sup>1</sup> apresentada ao Instituto de Ciências Matemáticas e de Computação, da Universidade de São Paulo - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.*

São Carlos  
Outubro de 2000

*“Nenhum trabalho, nenhuma tarefa é tão insignificante, tão simples, tão humilde, que não brilhe aos olhos de Deus”.*

*Portanto, seja qual a tarefa que você tem a desempenhar, receba-a e cumpra como se fosse a mais bela e mais importante.*

*Não dimensione a sua obrigação pelo que ela possa parecer diante da vaidade humana, mas pelo que possa valer diante da sua consciência.”*

*Este trabalho é dedicado a todos que contribuíram para a sua conclusão, e de modo especial, aos meus pais, aos meus irmãos e ao João.*

## AGRADECIMENTOS

A Deus, pela força e coragem que me foi concedida para enfrentar e vencer mais uma etapa da minha vida.

Aos meus pais, Nelma e Cornélio, pelo esforço dedicado à minha formação e pelo apoio, amor e carinho oferecidos gratuitamente. A vocês, o meu Obrigado!

Ao professor Dilvan, que acima de tudo é um grande amigo, por ter acreditado na minha capacidade de lutar e vencer. Obrigado pela confiança, pela oportunidade, pela ajuda nas horas de dúvidas, e principalmente, pela amizade.

Aos meus irmãos, Giovana e Alexandre, pelo carinho compartilhado em todos os momentos.

Aos meus avós, Pedro, Ivone, Nazareth e José Rocha, que sempre acreditaram e torceram por mim.

Ao João, pelo carinho, compreensão e amor dedicados durante estes quase três anos de convivência, e de modo especial, pela ajuda e apoio durante o mestrado. Obrigado por tudo, e principalmente, pelas lições ensinadas nos três anos de faculdade. Saiba que antes de tudo, você será, eternamente, o meu professor.

Às primas, Tânia e Inaiara, pela companhia e horas divertidas nos finais de semana de São Carlos e Araraquara.

A todos os meus familiares que torceram pela conclusão deste trabalho.

Ao colega Vanderley, companheiro de muito trabalho.

À psicóloga e amiga Patrícia, esposa do professor Dilvan, pela confiança, amizade e pela oportunidade de poder conviver com vocês. Obrigada também pelos conselhos em alguns momentos difíceis.

A todos os colegas que dividiram seu tempo comigo durante todo o mestrado, e em especial à Débora e à Íris, que foram duas grandes companheiras.

À Marília, Beth e Laura, pela paciência e por estarem sempre prontas a me atender.

À FAPESP (Fundação de Amparo à pesquisa do Estado de São Paulo), pelo apoio financeiro para a realização deste trabalho.

<b>ÍNDICE DE FIGURAS E TABELAS .....</b>	<b>vi</b>
<b>RESUMO .....</b>	<b>viii</b>
<b>ABSTRACT.....</b>	<b>ix</b>
 <b>1. INTRODUÇÃO .....</b>	 <b>1</b>
1.1. Considerações iniciais .....	1
1.2. Motivação .....	1
1.3. Objetivos .....	3
1.4. Estrutura.....	4
 <b>2. A EDUCAÇÃO A DISTÂNCIA E A INTERNET .....</b>	 <b>6</b>
2.1. Considerações iniciais .....	6
2.2. Os princípios da Educação a Distância.....	6
2.3. Evolução dos Programas de Educação a Distância .....	8
2.4. Evolução dos meios de comunicação .....	9
2.5. A Infra-estrutura da Internet .....	11
2.6. O ambiente WWW .....	12
2.7. Tecnologias Hipermídia.....	13
2.7.1. Multimídia e Hipermídia .....	13
2.7.2. Hiperdocumentos .....	15
2.7.3. HTML - Hypertext Markup Language.....	15
2.7.4. XML - Extensible Markup Language .....	17
2.7.5. XML - Novas Tecnologias .....	20
2.8. Considerações Finais .....	20
 <b>3. SISTEMAS DE APOIO À EDUCAÇÃO VIA INTERNET .....</b>	 <b>22</b>
3.1. Considerações iniciais .....	22
3.2. O ambiente WebCT.....	23
3.3. O ambiente AulaNet .....	26
3.4. O ambiente eClass .....	28
3.5. As ferramentas HyperBuilder, QuestBuilder e TaskBuilder .....	29
3.6. O ambiente StudyConf .....	31
3.7. Considerações finais .....	33
 <b>4. AGENTES DE SOFTWARE E A TECNOLOGIA JAVA .....</b>	 <b>34</b>
4.1. Considerações iniciais .....	34

4.2. O que é um agente? .....	35
4.3. Propriedade dos agentes .....	36
4.4. Classificação dos agentes .....	36
4.5. A linguagem JAVA e suas características .....	38
4.6. Desenvolvimento de aplicações distribuídas em Java .....	41
4.6.1. Serviços de comunicação via URL .....	41
4.6.2. Aplicações multithreading .....	42
4.6.3. Aplicações baseadas em comunicação via sockets .....	43
4.6.4. Aplicações para o acesso a bases de dados via JDBC .....	44
4.6.5. Java Beans .....	45
4.6.6. Java Servlets .....	46
4.6.7. Java Server Pages .....	48
4.7. Considerações finais .....	51
<b>5. MOBILE VIEWS .....</b>	<b>53</b>
5.1. Considerações iniciais .....	53
5.2. Conceituação das views de dados .....	53
5.3. O modelo de segurança das views .....	55
5.4. A mobilidade das views .....	58
5.5. Interação do usuário com as views .....	59
5.6. Considerações finais .....	60
<b>6. O WEBCOM .....</b>	<b>62</b>
6.1. Considerações iniciais .....	62
6.2. Visão geral do WebCoM .....	62
6.3. Arquitetura do WebCoM .....	64
6.4. Armazenamento de dados no WebCoM .....	67
6.5. Acesso ao WebCoM .....	68
6.6. As funcionalidades do WebCoM .....	70
6.6.1. A criação do ambiente .....	71
6.6.2. O gerenciamento das turmas de estudantes .....	72
6.6.3. O gerenciamento dos usuários .....	73
6.6.4. O gerenciamento das atividades didáticas .....	76
6.6.5. O gerenciamento das notas dos estudantes .....	80
6.6.6. A comunicação entre os usuários .....	81
6.7. Utilizando o WebCoM .....	82
6.8. Considerações finais .....	83
<b>7. CONCLUSÕES .....</b>	<b>84</b>
7.1. Considerações iniciais .....	84
7.2. Contribuições .....	84
7.3. Sugestões para Trabalhos Futuros .....	86
7.4. Considerações Finais .....	87
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>88</b>

## ÍNDICE DE FIGURAS E TABELAS

Figura 2.1 - Documentos estruturado de acordo com a linguagem HTML .....	16
Figura 2.2 - Apresentação do documentos estruturado de acordo com a linguagem HTML ....	16
Figura 2.3 - Exemplo de documento XML para um domínio de gerência de curso .....	18
Figura 3.1 - Interface de um curso de hipermídia no ambiente WebCT.....	24
Figura 3.2 - Interface pra transferência de arquivos no ambiente WebCT.....	25
Figura 3.4 - Interface principal de um curso oferecido no ambiente AulaNet .....	27
Figura 3.5 - Interface de uma aula no ambiente AulaNet .....	27
Figura 3.6 - (a) Sala de aula com recursos para capturar informação na sala de aula .....	29
Figura 3.6 - (b) Apresentação da informação capturada .....	29
Figura 3.7 - Interfaces da ferramenta TaskBuilder .....	30
Figura 3.8 - Interface da ferramenta QuestBuilder .....	31
Figura 3.9 - Interface da ferramenta HyperBuilder .....	31
Figura 3.10 - Página principal do StudyConf.....	32
Figura 3.11 - Página para realização de ações nos módulos aluno e administrador .....	33
Figura 4.1 - Esquema de compilação e execução de programas JAVA.....	39
Figura 4.2 - Anatomia básica de uma aplicação <i>stand-alone</i> .....	40
Figura 4.3 - Anatomia básica de um <i>applet</i> .....	41
Figura 4.4 - Esquema de comunicação cliente-servidor via socket .....	44
Figura 4.5 - Arquitetura do ambiente de passagem de mensagens do JDBC.....	45
Figura 4.6 - Arquitetura servlet baseada em primitivas send/receive .....	48
Figura 4.7 - Arquitetura genérica de utilização de JSP .....	50
Figura 4.8 - Exemplo de documento HTML que acessa página JSP .....	50
Figura 4.9 - (a) Documento HTML.....	51
Figura 4.9 - (b) Resultado gerado pela página JSP .....	51
Figura 5.1 - Interface View para implementação das mobile views .....	54
Figura 5.2 - Passagem de parâmetros para o applet Contact.....	55
Figura 5.3 - Interface de login para acesso às views .....	56
Figura 5.4 - Representação do modelo de segurança das views .....	57
Figura 5.5 - Representação do movimento de uma view .....	58
Figura 5.6 - View sendo apresentada na janela da ViewInterface .....	59
Figura 6.1 - Arquitetura geral do WebCoM.....	65
Figura 6.2 - Arquitetura dos agentes do WebCoM .....	66
Figura 6.3 - Organização das informações do WebCoM.....	67
Figura 6.4 - Organização das tabelas .....	68
Figura 6.5 - Página principal de acesso ao WebCoM .....	70

Figura 6.6 - Funções de cada usuário do WebCoM .....	70
Figura 6.7 - (a) Definição das informações básicas do curso.....	71
Figura 6.7 - (b) Definição da forma de avaliação do estudante (notas) .....	71
Figura 6.8 - (a) Definição dos dados da turma .....	71
Figura 6.8 - (b) Definição do tipo e número de atividades .....	72
Figura 6.9 - (a) Definição de uma atividade assignment .....	73
Figura 6.9 - (b) Definição de uma atividade report .....	73
Figura 6.10 - (a) Interface para cadastro de administrador para o curso.....	74
Figura 6.10 - (b) Interface para cadastro de monitores para o curso.....	74
Figura 6.11 - (a) Interface para cadastro de estudantes do curso .....	75
Figura 6.11 - (b) Interface para escolha de estudante a ser removido do curso .....	75
Figura 6.12 - Interface para aceitação de candidatos .....	75
Figura 6.13 - (a) Interface para visualização e alteração dos dados pessoais .....	76
Figura 6.13 - (b) Interface para alteração de senhas de usuários .....	76
Figura 6.14 - Interface para transferência de arquivos via applet .....	77
Figura 6.15 - (a) Interface para transferência de arquivos via protocolo HTTP .....	78
Figura 6.16 - (a) Interface para visualização e acesso às informações dos estudantes .....	79
Figura 6.16 - (b) Interface para visualização e acesso às informações dos grupos .....	79
Figura 6.17 - (b) Interface para manipulação de revisores de grupos de trabalho .....	80
Figura 6.18 - (a) Interface para atribuição de notas aos grupos de trabalhos.....	81
Figura 6.18 - (b) Interface para atribuição de notas aos estudantes individualmente .....	81
 Tabela 6.1 - Relação entre os cursos em que o WebCoM está sendo testado.....	 82

A aplicação da informática na educação tem sido alvo de intensas pesquisas nos últimos anos. Em ambientes de educação, o computador pode ser visto como um instrumento que facilita o acesso às informações, podendo ser considerado como um parceiro para as tarefas de orientação do professor. Adicionalmente, um crescente interesse tem sido observado em termos do desenvolvimento de aplicações, ferramentas e ambientes para suporte à produção e publicação de materiais didáticos via Internet - mais especificamente no ambiente WWW, criando uma variedade de cursos disponíveis neste ambiente. Genericamente, o processo para o oferecimento de cursos no ambiente WWW pode ser dividido em três etapas principais, que são a autoria, a disponibilização e o gerenciamento. Atualmente, as duas primeiras etapas podem ser realizadas com a utilização de ferramentas para autoria/disponibilização existentes. Por outro lado, a etapa de gerenciamento do curso ainda é uma tarefa pouco explorada. Assim, o objetivo deste trabalho está centrado na criação de ferramentas para ambientes de ensino distribuídos que promovam a interação entre estudantes e professores para o gerenciamento das atividades didáticas envolvidas em um curso, tais como formação de grupos de trabalho, entrega de trabalhos, visualização de notas, calendários, dentre outras. Este trabalho apresenta um conjunto de ferramentas – WebCoM (*Web Course Manager*) – baseado no conceito de agentes para a criação de um ambiente de gerenciamento e definição das atividades programadas para um curso via Internet. As ferramentas do WebCoM são fundamentadas no princípio do isolamento de professores e estudantes das tarefas de baixo nível do gerenciamento, como é o caso do acesso às bases de dados e ao volume de documentos gerados durante o curso.



The application of computers in education has been the target of intense research. In teaching environments, the computer can be viewed like an instrument to ease the access to information being considered like a partner for the teacher. A increasing interest has been observed in the development of applications, tools and programs for supporting the use of computers in teaching environments. These environments are normally based on the publication of teaching materials over the Internet, more specifically in the WWW environment. The process of offering courses in the WWW environment can be divided in three parts: material authoring, publication and management of the course. Nowadays, the first two steps can be done using many available tools, but the management step, however, isn't well supported by the tools currently available. Thus, the objective of this project is the development of tools to support the management tasks involved in Internet course, such as formation of work groups, delivery of assignments, visualization of grades, schedules, others. This work presents one set of tools – called WebCoM (Web Course Manager) – agents-based to the both creation of one management environment and definition of the scheduled activities for one course over the Internet. The WebCoM tools are based in the principle of the separation of both teachers and students of the lower level tasks of the management, like is the case of the access at the database and control on documents generated during the course.

### 1.1 Considerações iniciais

A aplicação da informática na educação tem sido alvo de intensas pesquisas devido à importância da utilização de ferramentas computacionais como apoio ao processo de ensino-aprendizagem. O uso da informática na educação constitui um novo paradigma, justificado por inúmeros autores que reforçam a aplicação dos computadores como agentes que contribuem para a construção do conhecimento (Santos Jr., 1998).

Inicialmente, o uso da informática na educação era visto como uma forma de ensinar a um grande número de pessoas dentro de um pequeno período de tempo. Atualmente, o uso dos computadores na educação é está centrado no potencial que a tecnologia agrega ao desenvolvimento de habilidades cognitivas essenciais para incrementar os processos de ensino, aprendizagem e treinamento.

O uso de computadores em ambientes educacionais tem sido incrementado coma evolução da Internet, que se apresenta como um fator importante a ser explorado em termos da disponibilização e consumo de materiais didáticos no ambiente WWW (*World Wide Web*) e de mecanismos de comunicação, tais como teleconferência e videoconferência (Lucena, 1997).

Neste capítulo são apresentados as motivações e os objetivos que levaram ao desenvolvimento deste trabalho e a forma como esta monografia está organizada.

### 1.2 Motivação

A proliferação dos computadores pessoais em conjunto com a popularização da Internet incrementou os níveis de comunicação entre os usuários, criando uma variedade de serviços oferecidos através desta rede heterogênea de ambientes, sistemas e plataformas. Serviços e aplicações de comércio eletrônico (e-Commerce), transações bancárias, entretenimento, *business*, aplicações educacionais, dentre outras, estão em franca expansão.

Apesar de todos os recursos oferecidos pelos computadores, o uso destes em ambientes educacionais, encontra, em muitos casos, resistências em virtude de um processo histórico envolvendo o relacionamento direto entre professor e estudante num ambiente caracterizado basicamente pelas salas de aula, lousa e giz. Estas resistências têm sido minimizadas através da conscientização dos professores no sentido de que o uso do computador pode ser um ponto importante, tanto como ferramenta de apoio quanto como instrumento facilitador do acesso à informação, sendo um parceiro para as funções de orientação do professor e não seu substituto (Valente, 2000).

A aplicação dos computadores em ambientes educacionais está dividida em duas categorias bem definidas (Barker, 1992):

- aprendizado assistido por computador (*Computer Aided Learning* - CAL) que focaliza o uso dos computadores como ferramenta para promoção do aprendizado, sendo visto como um agente que, inserido em ambientes de ensino-aprendizagem, promove uma forma de transmissão de conteúdos;
- treinamento baseado em computador (*Computer Based Training* - CBT) que focaliza o uso dos computadores como facilitadores do aprendizado de tarefas específicas em um determinado domínio de conhecimento, inclusive com o uso de simulações.

Segundo Valente (Valente, 2000), existem diferentes maneiras de usar o computador na educação. Uma maneira é informatizando os métodos tradicionais de instrução, que do ponto de vista pedagógico, seria o paradigma instrucionista. No entanto, o computador pode enriquecer ambientes de aprendizagem nos quais o estudante, interagindo com os objetos desse ambiente, tem chance de construir o seu conhecimento – paradigma construcionista. Nesse caso, o estudante não é mais instruído ou ensinado, mas é o construtor do seu próprio conhecimento.

O ambiente WWW da Internet tem sido reconhecido como um poderoso meio de disseminação de informação, principalmente por atrair um grande número de usuários, além de ser um canal de baixo custo, tanto para a produção quanto para o acesso a seus hiperdocumentos. A possibilidade de inserção de recursos multimídia nos documentos deste ambiente aumenta a sua relevância para o contexto da educação por incrementar o processo de aprendizagem do estudante através da utilização de imagens, sons, simulações e outros recursos (Keegan, 1991).

Em relação aos ambientes e ferramentas utilizados atualmente em ambientes de ensino-aprendizagem, pode-se observar uma concentração no desenvolvimento de sistemas de autoria/apresentação de material didático, sistemas para elaboração de exercícios, questionários e avaliações, sistemas tutores e de colaboração, dentre outros. Por outro lado, observa-se também uma carência de mecanismos específicos para gerenciamento e manipulação da informação produzida no desenvolvimento das atividades didáticas que fazem parte do processo de aprendizagem.

Alguns ambientes mais completos que oferecem suporte a estes tipos de atividades requerem, na maioria das vezes, a utilização do ambiente como um todo, criando uma espécie de dependência entre os processos de autoria do material e o gerenciamento das atividades. A necessidade de ferramentas específicas de gerenciamento reflete diretamente sobre os programas de educação a distância que necessitam de mecanismos especiais de coleta e disseminação de informações entre as entidades professor e estudante.

### **1.3 Objetivos**

Programas de educação a distância via Internet podem ser baseados na realização de cursos oferecidos através do ambiente WWW. O processo de oferecimento de um curso neste ambiente envolve, basicamente, a criação do curso e o seu gerenciamento.

A criação do curso passa pela etapa de autoria, no qual o professor e/ou autor cria o curso através da seleção de conteúdos; e pela disponibilização, em que o professor disponibiliza o curso através de um mecanismo de transferência de arquivos como o FTP (*File Transfer Protocol*). Algumas ferramentas baseadas no ambiente WWW permitem que a etapa de disponibilização seja eliminada, uma vez que o processo de autoria é realizado no próprio ambiente WWW. Uma vez disponível o material, torna-se necessário oferecer ferramentas apropriadas para apoiar a participação efetiva dos professores e estudantes durante a realização do curso, o que constitui a etapa de gerenciamento.

A etapa de gerenciamento de um curso pode ser vista sob dois aspectos diferentes: o gerenciamento sobre os materiais didáticos e conteúdos e o gerenciamento das atividades didáticas. O primeiro envolve o controle sobre os materiais didáticos, através, por exemplo, de mecanismos que monitoram o acesso pelo estudante, disponibilizando os conteúdos próprios

para cada estágio do curso. Já o gerenciamento das atividades didáticas envolve o controle sobre a definição e realização de todas as atividades que estão relacionadas ao curso, incluindo os trabalhos, os exercícios e as avaliações.

O objetivo deste trabalho concentra-se na etapa de gerenciamento das atividades didáticas, nos quais professores e estudantes podem manipular, de forma eficiente, as informações produzidas e coletadas pelas entidades envolvidas no curso.

Este trabalho focaliza a implementação de ferramentas que possam oferecer suporte à criação de um ambiente para o gerenciamento, via Internet, das atividades didáticas de um curso a distância. Através desse ambiente, o professor tem a possibilidade de acompanhar os resultados obtidos no desenvolvimento das tarefas pelos estudantes, por exemplo, no recebimento dos trabalhos, controle das propriedades do curso (data de entrega de trabalhos, disponibilização de projetos e notas dos estudantes), controle de estudantes cadastrados e senhas, dentre outras atividades.

Apesar de oferecer ferramentas para a interação dos estudantes com o curso, este trabalho está centrado na figura do professor com o intuito de minimizar a sua sobrecarga de trabalho quando da realização de um curso a distância.

## **1.4 Estrutura**

Esta dissertação está organizada de forma a apresentar o contexto teórico no qual este trabalho está inserido, bem como os resultados obtidos no desenvolvimento do trabalho e as suas contribuições para a comunidade interessada.

No **Capítulo 2**, são apresentados algumas características e princípios básicos da educação a distância, bem como uma breve descrição da evolução dos meios de comunicação e dos programas de educação à distância. Também são apresentadas algumas características da Internet, com vistas a contextualizá-la como um ambiente propício para o desenvolvimento dos programas mais recentes de educação a distância (última geração), além de apresentar um breve relato de tecnologias que têm sido freqüentemente utilizadas no desenvolvimento de sistemas e aplicações de suporte a educação via Internet, notadamente aquelas que usufruem do ambiente WWW.

No **Capítulo 3**, são relatados alguns ambientes e ferramentas que podem ser utilizados em educação a distância via ambiente WWW da Internet.

No **Capítulo 4**, são introduzidos os conceitos dos agentes de software, destacando-se algumas das definições encontradas na literatura, algumas propriedades dos agentes e uma classificação deste tipo de software. Os agentes de software são utilizados neste trabalho para implementar um processo de comunicação entre cliente e servidor. Além disto, é apresentada a linguagem Java como sendo uma tecnologia que tem revolucionado o processo de desenvolvimento de software, principalmente devido às suas características para a programação distribuída. Em termos da Internet, a linguagem Java tem se destacado como uma das principais linguagens de programação, e por isso foi escolhida e utilizada para o desenvolvimento das ferramentas deste trabalho, principalmente para o desenvolvimento dos agentes de software.

No **Capítulo 5** é apresentado o conceito de *mobile views* (*views* móveis), uma técnica utilizada neste trabalho para a implementação da maioria das ferramentas. A *mobile view* é uma forma eficiente de transferir dados através da Internet de forma segura, podendo ser utilizada para o desenvolvimento de várias aplicações baseadas na comunicação cliente/servidor via Internet. O conceito de *mobile views* está sendo proposto juntamente com as ferramentas deste trabalho.

No **Capítulo 6** apresenta-se o WebCoM (*Web Course Manager*) – nome dado ao conjunto de ferramentas desenvolvido no contexto deste trabalho – destacando-se as suas características principais, sua arquitetura e suas funcionalidades.

Finalmente, o **Capítulo 7**, apresenta as conclusões deste trabalho, considerando as dificuldades e contribuições, bem como algumas idéias para continuação e extensão deste trabalho.

## 2.1 Considerações iniciais

As constantes transformações culturais e tecnológicas requerem a evolução dos níveis de educação de capacitação para o trabalho. A tecnologia evolui em ritmo acelerado, permitindo a redução da distância entre instituições de ensino e os estudantes. Com uma metodologia adequada, os recursos tecnológicos podem, inclusive, melhorar a qualidade da educação presencial, com a utilização de meios de comunicação audiovisuais e de recursos da informática (Lucena, 1997).

O uso da Internet em ambientes educacionais apresenta-se como um novo paradigma a ser explorado em educação a distância. Muitas tecnologias têm sido propostas e estudadas, no sentido de aprimorar as técnicas existentes e permitir o desenvolvimento de aplicações cada vez elaboradas.

Neste capítulo é apresentada uma visão geral do termo “educação a distância” com vistas a situar o trabalho aqui reportado dentro deste contexto. A Internet é apresentada como um meio de distribuição de informação e de comunicação que oferece um ambiente propício para o desenvolvimento da educação a distância. Em adição, são apresentadas algumas tecnologias que têm sido difundidas e utilizadas na implementação dos mais variados sistemas de educação baseados na Internet.

## 2.2 Os princípios da Educação a Distância

As primeiras abordagens conceituais de educação a distância estabeleciam comparação imediata com a educação presencial, nos quais o professor, em sala de aula, é a figura central do processo ensino-aprendizagem. No Brasil, até hoje, muitas abordagens seguem essa linha conceitual, preferindo tratar a educação a distância a partir da comparação com a modalidade presencial de educação.

Embora esse comportamento não seja de todo incorreto, ele promove um entendimento parcial a respeito deste tipo de educação. Estudos mais recentes apontam para uma conceituação mais precisa do que é a educação a distância e como ela deve ser instituída e coordenada (Nunes, 1994).

Segundo Moore, a educação a distância é um método de instrução em que as condutas docentes acontecem à parte das discentes, de tal maneira que a comunicação entre o professor e o estudante possa ser realizada mediante textos impressos, por meios eletrônicos, mecânicos ou por outras técnicas (Moore et al., 1990).

Segundo Romiszowski, a educação/ensino a distância é um método racional de partilhar conhecimento, habilidades e atitudes, através da aplicação da divisão do trabalho, de princípios organizacionais, e pelo uso extensivo de meios de comunicação, especialmente para o propósito de reproduzir materiais técnicos de alta qualidade, os quais tornam possível instruir um grande número de estudantes ao mesmo tempo. É uma forma industrializada de ensinar e aprender (Romiszowski, 1993).

Na definição de Holmberg, o termo "educação a distância" esconde-se sob várias formas de estudo que não estão sob a contínua e imediata supervisão de tutores presentes com seus alunos nas salas de leitura ou num mesmo local. A educação a distância se beneficia do planejamento, direção e instrução da organização do ensino (Holmberg, 1977).

Em 1996, Moore e Kearsley apresentaram uma definição para educação a distância mencionando a importância dos meios de comunicação eletrônicos e a estrutura organizacional e administrativa específica, dizendo que a educação a distância é o aprendizado planejado que normalmente ocorre em lugar remoto e como consequência, requer técnicas especiais de planejamento de curso, técnicas instrucionais especiais, métodos especiais de comunicação (eletrônicos ou outros), bem como estrutura organizacional e administrativa específica (Moore & Kearsley, 1996).

Keegan sumariza os elementos que considera centrais, a partir dos conceitos acima enunciados (Keegan, 1991):

- separação física entre professor e estudante, que a distingue da educação presencial;



- influência da organização educacional (planejamento, sistematização, plano, projeto, organização dirigida), que a diferencia da educação individual;
- utilização de meios técnicos de comunicação para unir o professor ao estudante e transmitir os conteúdos educativos;
- previsão de uma comunicação bidirecional onde o estudante se beneficia de um diálogo com outro estudante ou professor;
- possibilidade de encontros ocasionais com propósitos didáticos e de socialização dos participantes.

Observando-se as várias definições da literatura, pode-se dizer que a educação a distância possui um vasto número de características que levam sempre a uma mesma idéia: a transposição das barreiras de distância ou de tempo existentes entre os professores e estudantes através da utilização dos mecanismos de comunicação.

A “educação a distância”, o “ensino a distância” e a “teleducação” são termos utilizados para expressar um mesmo processo de ensino-aprendizagem. Contudo, algumas pessoas ainda confundem teleducação como sendo somente educação por televisão, esquecendo-se que a palavra “tele” vem do grego “ao longe”, ou seja, “à distância”. Existem, ainda, diferenças entre educação a distância e educação aberta, porém prevalece, principalmente nos projetos universitários, forte ilusão de semelhança entre ambos os conceitos.

A educação aberta pode ser a distância ou presencial, distinguindo-se da educação tradicional em relação ao processo (materiais didáticos acessíveis por qualquer pessoa e métodos de aprendizagem controlados pelo estudantes), em relação ao insumo (não existem pré-requisitos acadêmicos e restrição social, econômica ou profissional), e em relação ao produto (os critérios de avaliação e aprovação são negociáveis e o conteúdo a ser estudado está sob controle dos estudantes) (Romiszowski, 2000).

### **2.3 Evolução dos Programas de Educação a Distância**

Segundo Romiszowski, a educação a distância pode ser dividida em quatro gerações que se diferem pela forma de instrução e pelo tipo de interação utilizada (Romiszowski, 2000).

A primeira geração corresponde ao Ensino por Correspondência que faz uso dos correios, para envio de materiais impressos aos estudantes, e de recursos mínimos de interação (interação lenta, inadequada e sem frequência) entre estudantes e professores.

A segunda geração – Teleducação - está ligada ao uso de meios de massa, tais como rádio, televisão e outros para a apresentação dos materiais aos estudantes. Nesta geração, a recepção dos materiais didáticos é organizada e controlada pelos instrutores.

A terceira geração é formada pelos Sistemas Integrados de Educação a Distância que usam múltiplos meios de instrução dos estudantes, incluindo os sistemas multimídia e multimeios (multimeios é a utilização de múltiplos meios de comunicação, tais como telefone, e-mail, fax, dentre outros, de forma integrada). A interação entre instrutores e estudantes é feita, geralmente, através de comunicação eletrônica por BBS (*Bulletin Board System*) ou via Internet.

Finalmente, a quarta geração é formada pelas Escolas Virtuais, que têm surgido nos últimos anos. Esta geração focaliza de forma especial o trabalho cooperativo e colaborativo via Internet entre os participantes e entre grupos de participantes do programa de educação. A interação, nesta geração, passa a ser síncrona (através de *chats*) ou assíncrona (através de grupos de discussão por *e-mail* e *net meeting*), além de poder ser em grupo ou individual.

Nota-se, através da evolução dos programas de educação a distância, relatados por Romiszowski (Romiszowski, 2000), que existe uma notável demanda em relação às formas de levar o conhecimento aos estudantes e em relação aos meios de interação que se tornam cada vez mais importantes, na medida em que a distância entre professores e estudantes aumenta. Atualmente, pode-se dizer que a Internet tem sido considerada como uma das principais formas de educar a distância, o que não significa necessariamente que esta será a última evolução neste sentido.

## **2.4 Evolução dos meios de comunicação**

O processo de comunicação entre os estudantes é fundamental para o sucesso de qualquer programa de educação, seja tradicional, dentro da sala de aula, ou a distância. A comunicação é um processo que visa, principalmente, a criação, comparação e compartilhamento de idéias entre um grupo de pessoas com interesses comuns (Romiszowski, 1993; Romiszowski, 2000).

A forma de comunicação mais conhecida e utilizada é a comunicação presencial que é altamente interativa e flexível, pois ambos os participantes do processo encontram-se face-a-face num mesmo local. Este tipo de comunicação depende da existência de uma linguagem comum entre os participantes e tem o poder de revelar estruturas conceituais na mente de cada um, além de poder ser reforçada pela utilização de gestos e expressões. A comunicação presencial é a forma mais rica de se comunicar.

Por outro lado, com a evolução da própria humanidade, e por consequência da tecnologia, surgiram formas alternativas de comunicação, não exigindo a presença face-a-face num mesmo local. Neste contexto, tecnologias como teleconferência e videoconferência têm sido utilizadas em vários domínios de aplicação, como é o caso da educação a distância.

Um ambiente de teleconferência torna possível o compartilhamento de um espaço acústico e visual através dos recursos das telecomunicações. Em uma teleconferência, os sinais de áudio e vídeo são gerados em um ponto de origem e difundidos por *broadcast* para um ou mais pontos de recepção, não sendo necessária a conexão entre estes pontos. Os participantes de uma sessão de teleconferência são capazes de visualizar mídias estáticas como textos, diagramas, figuras e documentos, sem recursos para compartilhamento automático dos mesmos. Assim, percebe-se que na teleconferência a interatividade é limitada (Lemair & Shae, 1997).

Em um ambiente de videoconferência, os sinais de áudio e vídeo são gerados por um ponto de origem e difundidos, mediante conexão prévia, para um ou mais pontos de recepção, caracterizando uma distribuição ponto-a-ponto. Os participantes de uma sessão de videoconferência devem dispor, por exemplo, de ferramentas que permitam a manipulação de documentos e mídias utilizados durante a sessão. Os participantes devem também ter oportunidade de comunicação, tanto em modo privado como em grupos (Lemair & Shae, 1997).

A comunicação mediada por computador (CMC) é uma outra forma de comunicação que compreende qualquer sistema de distribuição e compartilhamento de conteúdos, com ou sem recursos multimídia, através de computadores interligados em rede (geralmente, através da Internet). Exemplos típicos de CMC incluem o correio eletrônico, BBS, *chat*, *listserve*, grupos de apoio e jogos a distância. A união da CMC com os sistemas de videoconferências resultou nos sistemas de videoconferências *on-line*.

## 2.5 A Infra-estrutura da Internet

O número de serviços que podem ser disponibilizados na Internet é ilimitado, dada a transparência que o protocolo TCP/IP (*Transmission Control Protocol – Internet Protocol*) (Comer, 1995) oferece a essa rede, o que facilita o desenvolvimento contínuo de novas aplicações e serviços. O ponto comum entre esses serviços é o seu modelo de implementação - o modelo cliente/servidor de aplicações. Neste modelo, a execução do serviço é concentrada em programas servidores, enquanto o usuário acessa o serviço via programas clientes.

A interface com o usuário do lado cliente varia desde uma interface gráfica com operação via mouse (quando o equipamento onde reside o programa cliente tem esses recursos e acesso completo à Internet) até uma simples interface de texto que permita interação apenas via teclado (por exemplo, quando o programa cliente é acessado através de uma interface de terminal).

O lado servidor desse modelo pode ser implementado por um único programa ou por um conjunto de programas trabalhando em cooperação, como é o caso dos agentes de software, envolvendo inclusive sistemas de banco de dados.

Com base neste modelo de implementação, diversos serviços foram surgindo e se tornando conhecidos e utilizados pelos usuários da Internet. Dentre estes serviços, destaca-se o ambiente WWW (descrito na próxima seção) que é um sistema de busca e obtenção de informações no qual os caminhos de navegação são baseados nas técnicas de hipertexto. O ambiente WWW tem sido considerado como o maior responsável pelo crescimento da Internet nos últimos anos (Cyclades, 1996).

O serviço de correio eletrônico (e-mail) é considerado como o de maior alcance da Internet, pois permite a troca de mensagens com quaisquer usuários de redes que estejam conectadas à Internet.

O serviço de *Network News* – Serviço de BBS – (*Usenet News* ou *News*) é formado por informações agrupadas por categorias e por programas responsáveis pelo seu intercâmbio, divulgação e acesso. As categorias em que as informações (ou assunto) são agrupadas são denominadas como *newsgroups*, organizadas de forma hierárquica. Os usuários podem participar

dos grupos de interesse para a simples leitura dos artigos (uma unidade do *newsgroups*) ou, até mesmo, para o envio de artigos próprios ou respostas a outros artigos.

O FTP é o serviço padrão para a transferência de arquivos, serviço este que consiste em obter ou enviar arquivos para outros computadores da Internet. Ao lado do FTP está a execução remota de aplicações – Telnet, que permite a execução de programas em outros equipamentos localizados remotamente via Internet. Ambos estes serviços são baseados no processo de *login* (autorização) para a sua utilização.

Além do ambiente WWW, a Internet conta com outro serviço para procura de informações – Gopher – que apresenta uma dada informação (textos, imagens, dentre outras) através de menus.

Todos estes serviços, apesar de não terem sido projetados exclusivamente para este fim, tem sido bastante difundidos e utilizados em ambientes de educação, especialmente a distância. No **Capítulo 3** apresenta-se alguns exemplos de ambientes e ferramentas que oferecem suporte à educação, que certamente fazem uso de um ou mais destes serviços oferecidos através da Internet.

## 2.6 O ambiente WWW

Por muitos anos, as pessoas “sonharam” com o conceito de uma base de dados universal de informação que pudesse ser acessada por qualquer pessoa em qualquer parte do mundo a um baixo custo. Atualmente, o ambiente WWW tem tornado possível a realização deste “sonho”.

O objetivo inicial da proposta do ambiente WWW era a troca de informações através de hipertextos (documentos de texto com *links* de um documento para outro). Posteriormente, as potencialidades de troca de áudio, vídeo e imagem foram sendo agregadas aos documentos, aumentando o interesse pela publicação de materiais das mais diversas áreas do conhecimento. Atualmente, o principal objetivo do ambiente WWW é prover aos usuários de redes de computadores o acesso simplificado a uma vasta variedade de informações multimídia e hipermídia, através da utilização de softwares com interfaces comuns e populares, tais como o Netscape (Netscape, 1998), Internet Explorer (Microsoft, 1999), dentre outros (Cyclades, 1996).

A popularidade do ambiente WWW está centrada na simplicidade da interface, facilidade de publicação de documentos e baixo custo. Em adição, a possibilidade de utilização de recursos hipermídia complementa as características que atraem o interesse da maioria dos usuários.

O processo de utilização do ambiente WWW inicia-se quando um *browser*, executando em uma máquina cliente, conecta-se a um servidor através do fornecimento de um endereço URL (*Uniform Resource Location*) e, utilizando o protocolo HTTP (*Hypertext Transfer Protocol*) (HTTP, 1992), acessa um documento HTML (*HyperText Markup Language*) transferindo-o para ser visualizado na máquina cliente. Estando o documento na máquina cliente, um *parser* analisa o código fonte, processa a interpretação do documento e gera o modelo gráfico para apresentação. Após a apresentação do documento, o usuário pode então “navegar” pelas suas informações e também acionar *links* para outros documentos, caso existam.

O crescente interesse pela publicação de materiais e a necessidade de aplicações mais complexas levaram à realização de pesquisas no sentido de expandir as potencialidades do ambiente WWW, seja pela extensão da linguagem HTML ou pela criação de novas tecnologias, com vistas a fornecer novos recursos para a implementação de aplicações. No contexto da educação, o ambiente WWW se destaca pelo seu alto potencial de armazenamento/apresentação de informações incrementado pela utilização dos recursos multimídia.

## **2.7 Tecnologias Hipermídia**

O uso dos computadores nos mais diversos segmentos da sociedade e a expansão da Internet são alguns dos fatores que têm impulsionado a pesquisa e o desenvolvimento de tecnologias computacionais cada vez mais elaboradas e próximas ao cotidiano das pessoas. Neste contexto, torna-se relevante a apresentação de algumas dessas tecnologias que têm sido utilizadas no desenvolvimento de ambientes e ferramentas voltados para aplicações em educação.

### **2.7.1 Multimídia e Hipermídia**

Nos últimos anos tem-se observado uma explosão de interesse em multimídia, hipertextos e sistemas hipermídia. Literaturas e seminários têm sido realizados no sentido de definir “o que é necessário para desenvolver sistemas hipermídia e hipertextos”, o que ressalta a importância desta área no atual contexto tecnológico.

Um sistema multimídia é aquele no qual documentos podem ser compostos por diferentes tipos de mídia. Um sistema hipermídia é definido como um sistema multimídia que também possui *hyperlinks* entre seus documentos, sendo que um sistema hipermídia é basicamente um sistema hipertexto que incorpora um significativo número de tipos de mídias na apresentação de informações ao usuário, tais como textos, gráficos, imagens, áudio, vídeo, animações, dentre outros (Bufford, 1994).

Um sistema multimídia é uma coleção de componentes de hardware e software relacionados entre si e que devem ser selecionados de forma a funcionarem juntos. Esse conjunto de componentes é formado, basicamente, pela placa de vídeo, barramento de dados, controladoras, software de aquisição, software de edição, software de autoria e/ou programação e mídias de distribuição.

No contexto da hipermídia, potencialmente, observa-se que o desenvolvedor pode usufruir de recursos que permitam a sofisticação das aplicações. Neste ponto, existem algumas regras que podem ser usadas por usuários e desenvolvedores para nortear e avaliar sistemas hipermídia, tais como escalabilidade, interoperabilidade, acesso multiusuário, dentre outras (Harrison, 1996).

No contexto dos sistemas distribuídos, um sistema hipermídia suporta a construção cooperativa de aplicações por múltiplos autores e a execução dessas aplicações e apresentação de informações a múltiplos usuários em redes de computadores, geralmente formadas por componentes heterogêneos.

Os sistemas hipermídia devem suportar ambientes cliente-servidor, nos quais objetos de uma aplicação podem ser acessados por outros usuários, sendo que o sistema deve resolver conflitos quando dois ou mais usuários tentarem acessar e modificar um mesmo objeto. Deve-se considerar, ainda, que não somente a aplicação e o processamento devem ser distribuídos, mas também o armazenamento e o gerenciamento, o que exige a implementação de mecanismos que garantam transparência e permitam ao usuário utilizar a rede como se fosse o seu próprio equipamento (Collouris et al., 1994).

### 2.7.2 Hiperdocumentos

No contexto do ambiente WWW, um hipertexto é um documento formatado com o uso de uma linguagem de *markups*, como HTML, que pode ser visualizado pelo usuário através de um navegador (*browser*). Ainda utilizando uma linguagem de *markups*, pode-se estabelecer ligações (*links*) entre documentos, caracterizando um hiperdocumento.

Nos hiperdocumentos podem ser apresentados textos formatados, imagens estáticas e em movimento (vídeos), além de tabelas de dados e sons, que são características importantes para ambientes de educação. Os hiperdocumentos são ditos estruturados devido ao significado semântico de seus elementos, como o significado entendido a partir da *tag* <TITLE> que é o título de uma página (embora um documento formatado com a linguagem HTML possa não ser totalmente estruturado, como apresentado na próxima seção). Um bom exemplo para a formatação de documentos estruturados é o padrão SGML (*Standard Generalized Markup Language*) (ISO, 1986) e a linguagem XML (*Extensible Markup Language*) (Connolly, 1997).

### 2.7.3 HTML - Hypertext Markup Language

A linguagem SGML permite que documentos armazenados eletronicamente sejam definidos em termos de seu conteúdo e sua estrutura, independentemente de sua forma de apresentação (ISO, 1986). Em adição, a linguagem SGML tem flexibilidade para definir um conjunto ilimitado de linguagens específicas, o que define SGML como meta-linguagem.

HTML é uma linguagem, criada a partir da meta-linguagem SGML, que se preocupa principalmente em organizar como um documento deve ser apresentado. Além disso, HTML surgiu devido à expansão da Internet e à necessidade de uma linguagem simples para especificação da forma de apresentação dos documentos no ambiente WWW (HTML, 1992; HTML, 1997).

A linguagem HTML possui *tags* (etiquetas) pré-definidas que são utilizadas para especificar a informação e a forma como esta informação deve ser apresentada, tais como <HEAD>, para definir o cabeçalho do hiperdocumento; <P>, para definir um parágrafo; e <TABLE>, para definir uma tabela. As **Figuras 2.1** e **2.2** apresentam um documento formatado com a linguagem HTML e o resultado da apresentação gráfica desse mesmo documento no *browser*.



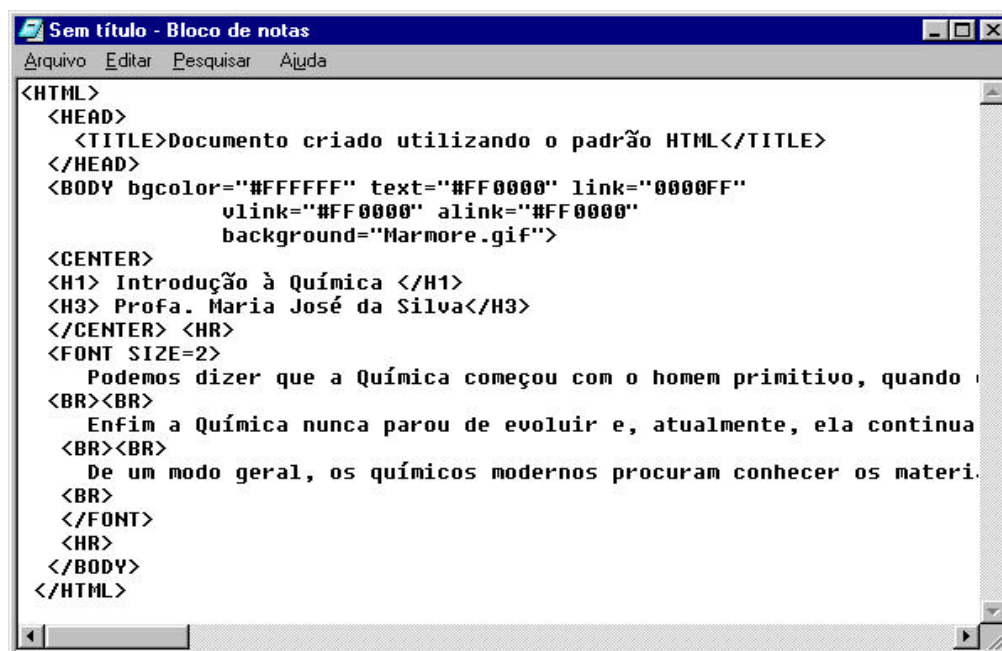


Figura 2.1 – Documento estruturado de acordo com a linguagem HTML

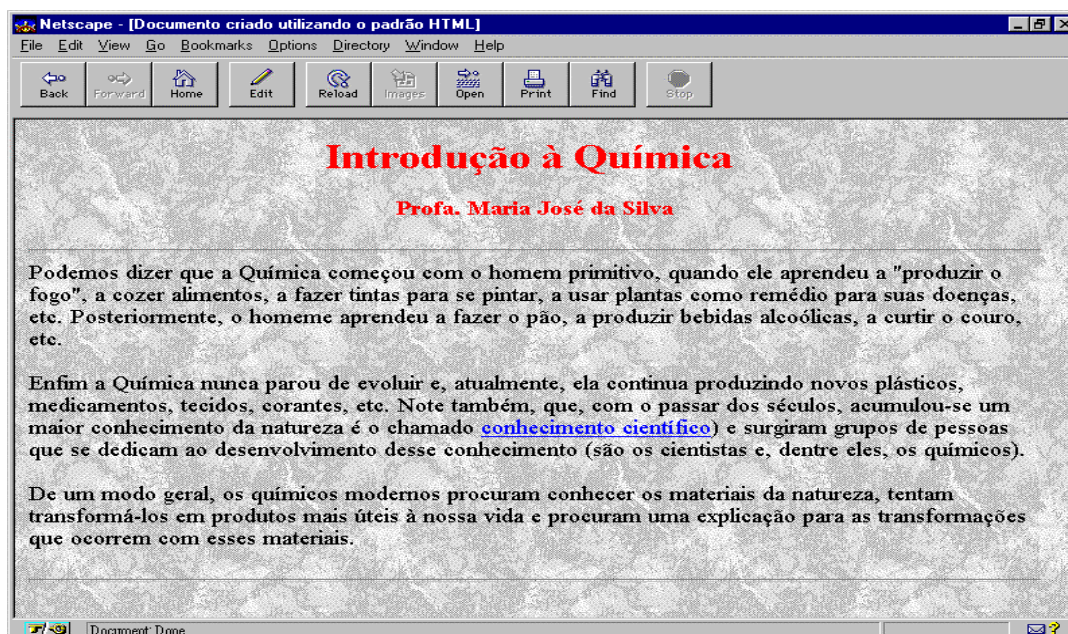


Figura 2.2 – Apresentação do documento estruturado de acordo com a linguagem HTML

A linguagem HTML apresenta algumas limitações relacionadas à sua flexibilidade (Bray et al., 1997; Johnson, 1999):

- HTML não permite a especificação de novos elementos e atributos;
- HTML representa a informação em termos de seu *layout*;

- a alteração de um documento HTML é trabalhosa, visto que é necessário alterar o documento como um todo (apresentação e conteúdo).
- HTML possui pouca estrutura semântica, pois seus elementos são agrupados sem seguir uma estrutura pré-definida.

Apesar destas características, HTML é uma linguagem de singular importância, pois a simplicidade do código e a limitação no número de elementos que a constitui permitem que o seu uso seja simplificado e que qualquer pessoa com conhecimentos básicos de computação possa implementar um hipertexto para ser apresentado no ambiente WWW.

Para suprir algumas limitações da linguagem HTML, novas tecnologias têm sido propostas, como é o caso de DHTML (*Dynamic HTML*). Numa perspectiva inovadora, as páginas desenvolvidas com recursos DHTML ganham “vida” com a utilização de *scripts* simples e fáceis de desenvolver (por exemplo, em JavaScript), que tornam dinâmicos os elementos de um documento HTML. A linguagem principal do ambiente WWW.

#### 2.7.4 XML - Extensible Markup Language

Outra linguagem definida a partir de SGML é a linguagem XML (Connolly, 1997) que tem por objetivo separar o conteúdo de um documento de sua forma de apresentação.

Como o próprio nome sugere, XML é uma linguagem extensível (meta-linguagem), com a qual os projetistas podem criar seus próprios elementos de acordo com a aplicação que está sendo modelada, dando importância ao conteúdo e à estrutura da informação, sem se preocupar com a apresentação. Através desta característica, a flexibilidade da linguagem é ampla, uma vez que o projetista pode criar uma nova linguagem a partir da meta-linguagem XML. Para ilustrar, a **Figura 2.3** apresenta um documento estruturado de acordo com a linguagem XML.

Neste exemplo nota-se que não há nenhum elemento que define como as informações contidas no documento XML devem ser apresentadas, diferente do que acontece em um documento formatado com a linguagem HTML. Sendo assim, pode-se dizer que XML considera apenas o conteúdo a ser apresentado, o que cria a necessidade da utilização de outro recurso que seja responsável pelos atributos de apresentação. Esta necessidade pode ser suprida através da utilização de programas específicos ou linguagens apropriadas para associar estilos ao conteúdo

de um documento XML, como é o caso da tecnologia XSL (*Extended Style language*) (Johnson, 1999).

A interpretação de um documento XML ocorre através da utilização de um *parser*, que analisa o documento obtendo suas informações de conteúdo para processamento. Esta função depende do tipo de *parser* implementado, do propósito da informação e da estrutura semântica apresentada no documento XML.

```
<GERÊNCIA_CURSO>
  <CURSO>
    <NOME_CURSO> Sistemas Operacionais </NOME_CURSO>
    <PROFESSOR> Dilvan de Abreu Moreira</PROFESSOR>
    <ALUNOS>
      <ALUNO>
        <IDENTIFICAÇÃO_CURSO> 003 </IDENTIFICAÇÃO_CURSO>
        <NOME> José da Silva </NOME>
        <MATRICULA> 596333 </MATRÍCULA>
        <EMAIL> jose@icmc.sc.usp.br </EMAIL>
        <HOMEPAGE> http://www.icmc.sc.usp.br/~jose </HOMEPAGE>
        <NOTAS>
          <NOTA> 5.0 </NOTA>
          <NOTA> 4.0 </NOTA>
          <NOTA_FINAL> 4.5</NOTA_FINAL>
        </NOTAS>
      </ALUNO>
      <ALUNO>
        <IDENTIFICAÇÃO_CURSO> 003 </IDENTIFICAÇÃO_CURSO>
        <NOME> Maria do Carmo </NOME>
        <MATRICULA> 596354 </MATRÍCULA>
        <EMAIL> maria@icmc.sc.usp.br </EMAIL>
        <HOMEPAGE> http://www.icmc.sc.usp.br/~maria </HOMEPAGE>
        <NOTAS>
          <NOTA> 9.0 </NOTA>
          <NOTA> 9.0 </NOTA>
          <NOTA_FINAL> 9.0</NOTA_FINAL>
        </NOTAS>
      </ALUNO>
    </ALUNOS>
  </CURSO>
</GERÊNCIA_CURSO>
```

Figura 2.3 – Exemplo de documento XML para um domínio de gerência de curso

Cada documento XML pode ou não estar associado a uma linguagem específica que define sua estrutura. Essa definição é formalizada através da criação de um DTD (*Document Type Definition*). De modo geral, um DTD define regras para a especificação de uma classe de documentos, tais como (Jonhson, 1999):

- que tipos de elementos podem existir em um documento (por exemplo, Aluno, Professor, Curso);

- que atributos esses elementos podem ter (por exemplo, um Curso é formado pelos elementos Nome\_Curso, Professor, Período, Alunos);
- como as instâncias desses elementos estão hierarquicamente relacionadas (como por exemplo, um Curso possui Alunos e cada Aluno possui Identificação\_Curso, Nome, Email, Homepage e Notas, que por sua vez podem conter várias Notas e uma Nota\_Final).

A estrutura especificada em um DTD, segundo sua definição no padrão SGML, possui uma propriedade importante: apenas a estrutura lógica de um documento é descrita, não sendo fornecida informação sobre o conteúdo e apresentação dos elementos definidos (Brown, 1989).

Se o documento XML estiver associado a um DTD, então o *parser* deve validá-lo (*parser* de validação), processando o DTD que corresponde ao documento XML e obtendo sua estrutura. Posteriormente, o *parser* obtém as informações do documento XML realizando a validação com a estrutura definida no DTD. Por outro lado, se o documento XML não está associado a um DTD, então apenas a estrutura sintática do documento XML é verificada (*parser* sem validação).

De forma associada ao desenvolvimento da tecnologia XML, o grupo responsável desenvolveu também uma linguagem de apresentação – XSL (*Extended Style Language*), que é utilizada para formatar o estilo de apresentação do conteúdo de um documento XML (Jonhson, 1999).

A especificação de XSL está dividida em duas partes: a primeira tem por objetivo “transformar” documentos XML em outros formatos de documentos, tais como HTML, TeX, *PostScript* e RTF, e é chamada de XSLT (*XSL Transformation*). A outra parte está voltada para a formatação de estilos de apresentação de documentos XML, que possibilita a criação de múltiplas representações da mesma informação através do uso de vários documentos XSL diferentes aplicados a um único documento XML.

Ainda no contexto de XML, outra tecnologia tem sido desenvolvida – a linguagem XQL (*XML Query Language*) - que é uma notação para filtragem de informações de um documento XML (Robie et al., 1998). A linguagem XQL provê mecanismos de filtragem de informações, e os resultados obtidos podem ser utilizados em diversos contextos em uma aplicação. A notação da linguagem XQL é simplificada e foi projetada especificamente para documentos XML.

### 2.7.5 Novas tecnologias

No contexto da Internet, tecnologias são criadas, transformadas e melhoradas constantemente, e com potencial para o desenvolvimento de aplicações hipermídia, como é o caso daquelas voltadas para ambientes de educação.

Notadamente, algumas dessas tecnologias merecem ser referenciadas no contexto mais amplo no qual este trabalho se insere: a educação a distância.

Tecnologias como SMIL (*Synchronized Multimedia Integration Language*) (SMIL, 1999), VRML (*Virtual Reality Modeling Language*) (Burdea & Coffet, 1993), MPEG-4 e MPEG-7 (ISO, 1997) podem ser exploradas tanto para o desenvolvimento de conteúdos didáticos para educação a distância quanto para o desenvolvimento de ferramentas para autoria, disponibilização e gerenciamento de cursos.

### 2.8 Considerações Finais

A educação a distância é um campo que tem crescido muito nos últimos anos, especialmente pela democratização do acesso à informação. As ferramentas utilizadas na educação a distância são, em muitos casos, extensões e/ou adaptações de ferramentas utilizadas na educação presencial incrementadas por recursos computacionais e pela Internet. A educação a distância permite que estudantes dispersos no tempo e no espaço interajam entre si e com o professor no processo de aprendizagem, permitindo que cada estudante organize seu próprio horário de estudo. Apesar das facilidades da educação a distância, deve-se também ter em mente as suas limitações que podem, algumas vezes, inviabilizar a sua aplicação e/ou seu sucesso.

Conforme apresentado, novas tecnologias têm sido criadas e, gradativamente, vão sendo incorporadas às aplicações, em especial na implementação de aplicações para o ambiente WWW. O uso de hiperdocumentos, tanto em HTML quanto em XML, permite a apresentação controlada de conteúdos a grupos de usuários, o controle do uso do conteúdo pelo usuário, bem como o reuso de classes de documentos em mais de uma aplicação, o que são fatores essenciais em programas de educação, especialmente a distância.

A separação entre estudantes e professores em programas de educação a distância leva à necessidade de mecanismos especiais de coleta, armazenamento e compartilhamento das

informações produzidas no desenvolvimento de atividades didáticas de um curso. A proposta deste trabalho é fornecer suporte a esta necessidade utilizando a própria Internet como ambiente para gerenciamento dos cursos oferecidos. Os próximos capítulos apresentam o contexto mais específico no qual este trabalho se insere.

# 3

## SISTEMAS DE APOIO À EDUCAÇÃO VIA INTERNET

### 3.1 Considerações iniciais

Com base nas tecnologias e nos serviços oferecidos pela Internet, diversos trabalhos de aplicação das tecnologias da informação na educação têm sido propostos e implementados. Alguns destes ambientes, apesar de não terem sido projetados para este fim, são utilizados como ferramentas de apoio aos programas de educação a distância.

Atualmente, os autores de material didático para o ambiente WWW têm à sua disposição diversos meios de autoria/publicação de documentos que, apesar de exigirem algum conhecimento especializado em informática (especialmente em Java e tecnologias afins), são muito poderosos e versáteis. Um modelo básico de produção de material multimídia para educação via Internet pode ser:

- codificação, através das linguagens de *markups* ou ambientes e ferramentas de autoria, dos documentos que fazem parte do curso a ser oferecido no programa de educação;
- inserção, se necessário, de mecanismos de simulações e interações do usuário em tempo de apresentação dos documentos, através da implementação de *applets* Java e/ou *scripts*;
- disponibilização do material didático em um servidor de documentos HTML através de um mecanismo de transferência de arquivos, tal como o FTP, para posterior apresentação ao usuário final através do protocolo HTTP e um *browser*.

Após a autoria/disponibilização do material didático, ainda é necessário o gerenciamento do curso.

De modo geral, existem ferramentas apropriadas para os vários estágios de criação de material didático com ou sem multimídia. Estas ferramentas fornecem o apoio necessário para facilitar o processo de autoria e disponibilização do material didático e podem ser simples e fáceis ou complexas e muito poderosas.

Como exemplos comerciais, tem-se o FrontPage da Microsoft (Microsoft, 1999), o Composer da Netscape (Netscape, 1998), dentre outras.

No mundo acadêmico, muitos esforços vêm sendo realizados no sentido de explorar o uso de recursos computacionais em ambientes de educação. É importante ressaltar que várias ferramentas têm sido implementadas para apoiar a elaboração do material didático, observando-se, principalmente, a autoria pelo professor e a navegação do estudante pelo conteúdo do material elaborado.

Como exemplos de aplicações educacionais, pode-se citar o ambiente **WebCT**, desenvolvido na University of British Columbia, Canadá, em 1996, destinado a facilitar a criação de ambientes sofisticados de educação no ambiente WWW (WebCT, 1999). Existem também os ambientes AulaNet (AulaNet, 1998), TopClass (TopClass, 1999), StudyConf (Macedo et al., 1999), eClass (Abowd, 1999), e várias ferramentas de autoria, tais como HyperBuilder, QuestBuilder e TaskBuilder (Santos Jr, 1998), WebCourse (Scapin & Garcia Neto, 1997), dentre outras. Outras referências podem ser encontradas em (Nunes et al., 1997; Moreira et al., 1995; Freire & Nunes, 1999).

Neste capítulo são apresentados alguns ambientes e ferramentas, relevantes para o contexto deste trabalho, que oferecem apoio à educação via Internet. Conforme citado, através da possibilidade de criação e gerenciamento de cursos, alguns destes ambientes e/ou ferramentas têm sido freqüentemente utilizados como suporte a programas de educação a distância.

### 3.2 O ambiente WebCT

O ambiente WebCT é baseado em documentos HTML tanto para o estudante quanto para o autor, e é apresentado através de um documento HTML principal com *links* para conteúdo dos cursos, propriedades e ferramentas.

Atualmente, o WebCT é visto como um dos mais completos ambientes para desenvolvimento de cursos para o ambiente WWW, por fornecer aos estudantes e professores os recursos necessários para a realização do curso, incluindo as ferramentas de gerenciamento. A **Figura 3.1** apresenta a interface inicial de um curso criado com a utilização do WebCT.



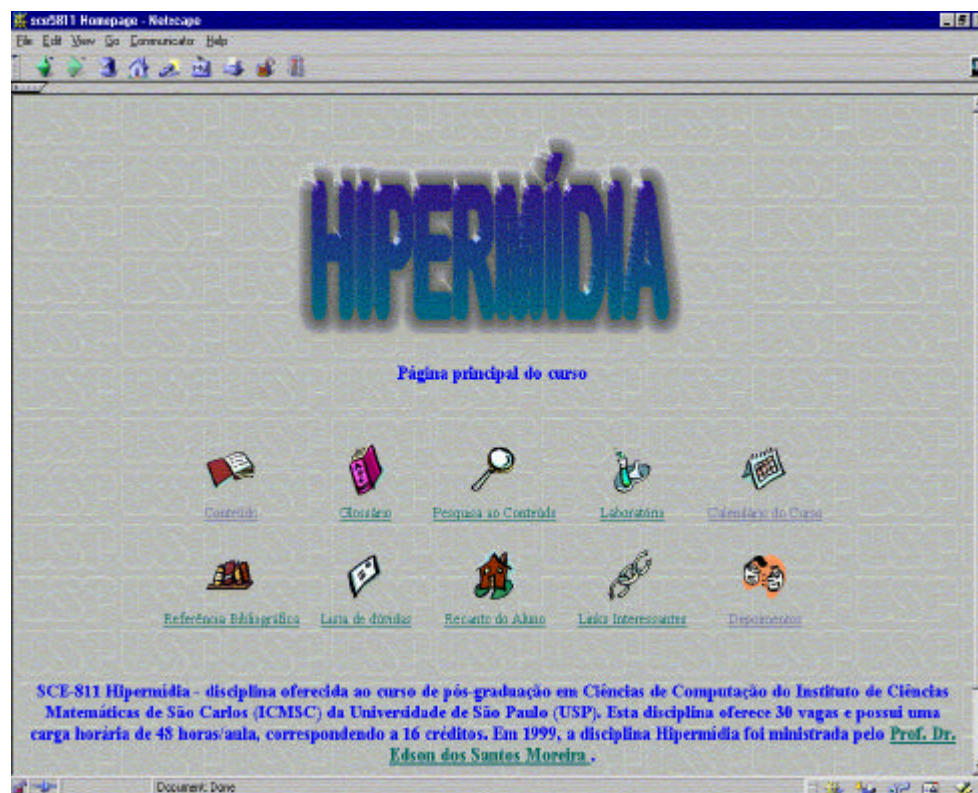


Figura 3.1 - Interface de um curso de hipermídia no ambiente WebCT

WebCT existem quatro grupos de usuários que possuem direitos de acesso diferentes sobre as ferramentas do ambiente, que são (WebCT, 1999):

- **Administrador:** é o usuário responsável pelo servidor WebCT (onde ficam armazenadas todas as informações sobre os cursos disponíveis); é quem autoriza a criação de novos cursos e cria os usuários *designers*;
- **Designer:** na maioria dos casos, é o professor responsável por um curso; é quem cria e organiza as páginas do curso, publica na Internet enviando para o Servidor WebCT e também cadastra os usuários alunos e lhes atribui notas, além de poder criar o usuário monitor;
- **Monitores:** usuários que ajudam os professores a administrar os usuários alunos e suas notas; eles podem criar, alterar ou excluir usuários alunos do curso;
- **Alunos:** são os usuários que realizam o curso criado; cada curso tem um grupo de usuários alunos que foi cadastrado por um *designer* ou por um monitor.

O ambiente WebCT oferece ferramentas para autoria do material didático e para o gerenciamento da maioria das atividades, tais como o fornecimento de dados estatísticos do

curso, progresso individual do estudante, controle de acesso pelos estudantes, testes *on-line* com tempo pré-determinado e correção automática dos testes, dentre outros. As **Figuras 3.2** e **3.3** apresentam interfaces do usuário *Designer* para transferência de arquivos (*upload* para o servidor do curso) e para gerenciamento de estudantes do curso.

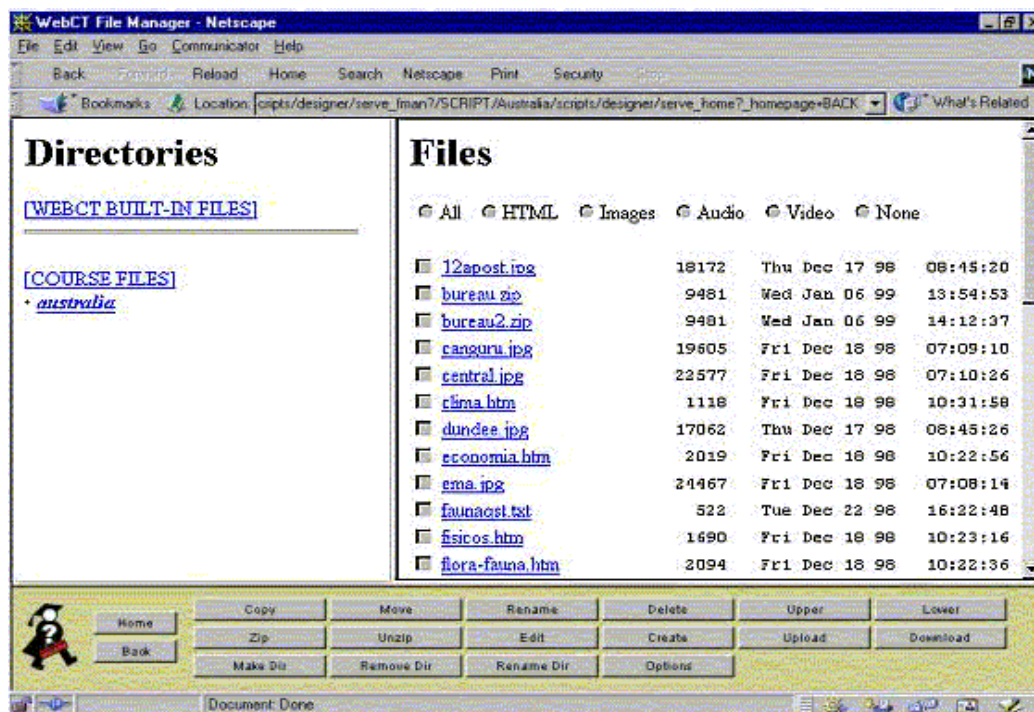


Figura 3.2 – Interface para transferência de arquivos no ambiente WebCT

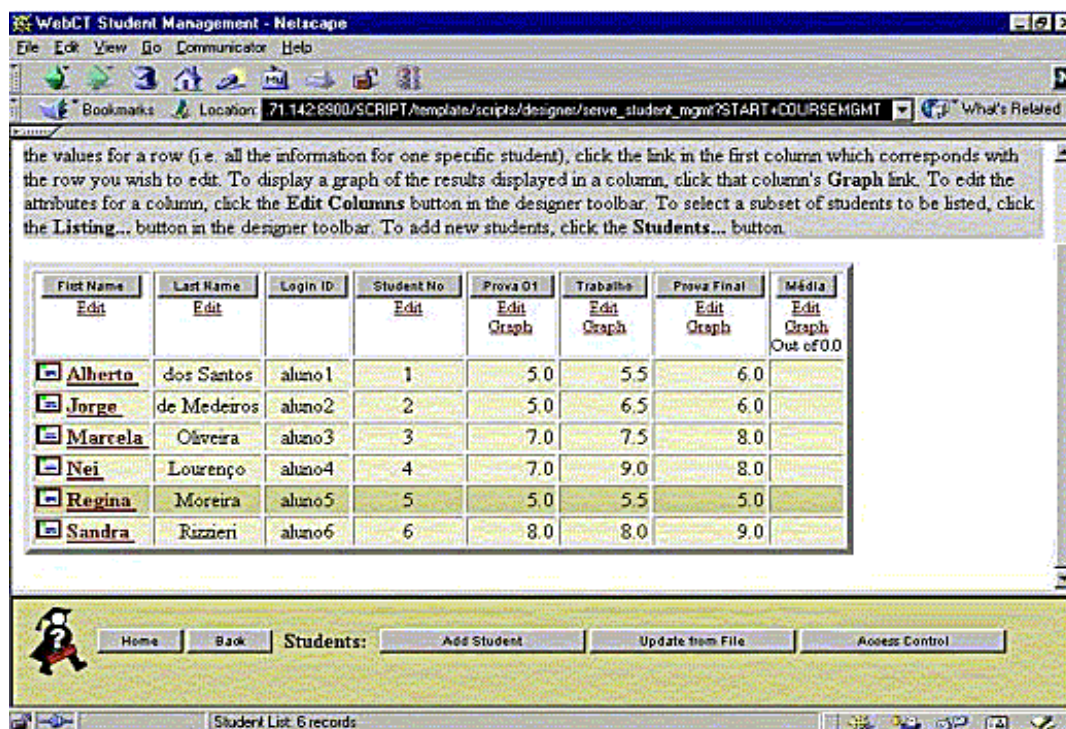


Figura 3.3 – Interface para gerenciamento de estudantes no ambiente WebCT

### 3.3 O ambiente AulaNet

Além do ambiente **WebCT**, o ambiente **AulaNet** é um bom exemplo para autoria e disponibilização de material didático para o ambiente WWW. Desenvolvido no Laboratório de Informática da PUC-RIO, é um ambiente para criação de cursos direcionados ao público leigo. O ambiente AulaNet visa adotar a *Web* como ambiente educacional, criando uma transição da sala de aula convencional para a sala de aula virtual, oferecendo a oportunidade de se reusar o material educacional existente. Em adição, a proposta do ambiente Aulanet é permitir um elevado grau de interatividade e intensa participação do estudante, sem que o autor tenha conhecimento profundo sobre o ambiente WWW (Aulanet, 1997).

Basicamente, o processo de criação de um curso através do ambiente Aulanet inclui a identificação do ator (que pode ser o autor, o estudante e o administrador); a descrição das informações gerais do curso (nome, descrição sumária, e ementa); e a seleção de recursos (que podem ser didáticos, de avaliação, administrativos e fixos). O administrador é caracterizado como sendo a pessoa que cuida das tarefas de gerenciamento, tais como inscrição do estudante, divulgação da agenda e outras.

Explorando ferramentas *freeware* disponíveis na Internet, o AulaNet provê suporte a uma série de recursos didáticos (Grupo de Interesse, Grupo de Discussão e Debate) que são mapeados para serviços AulaNet suportados por ferramentas associadas (Fucks & Lucena, 2000).

Para o acesso ao curso é necessário que o estudante esteja cadastrado no ambiente e que sua matrícula seja solicitada. A **Figura 3.4** apresenta a interface principal de um curso oferecido no ambiente AulaNet, onde se observa que a barra de ferramentas, com os serviços selecionados (para o curso) pelo professor, é apresentada em uma nova janela do *browser*.

A **Figura 3.5** apresenta o exemplo de uma aula que contém recursos de vídeo, transparência e texto de aula. Nesta figura, observa-se que há a divisão da janela do *browser* em duas áreas, uma para a apresentação da transparência (à esquerda) e outra para o texto de aula (à direita). Em adição, duas outras janelas são necessárias: a janela com a barra de ferramentas e a janela do vídeo que apresenta os controles necessários à execução do vídeo (AulaNet, 1998).



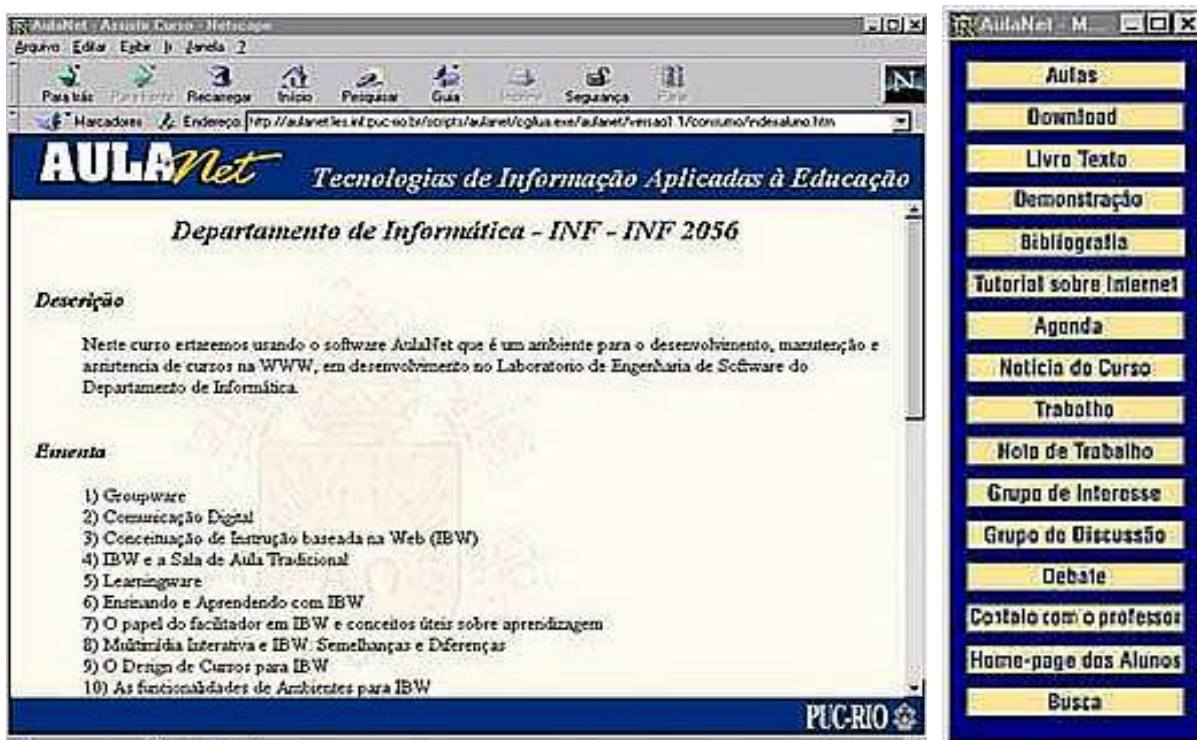


Figura 3.4 – Interface principal de um curso oferecido no ambiente AulaNet (Aulanet, 1998)



Figura 3.5 – Interface de uma aula no ambiente AulaNet (Aulanet, 1998)

Os serviços oferecidos no ambiente AulaNet estão divididos em três categorias: serviços de comunicação, cooperação e coordenação. Os serviços de comunicação compreendem os contatos com o docente, grupos de discussão, grupos de interesse, debates e contatos com os participantes do curso. Os serviços de coordenação envolvem as agendas, planos de aulas, avaliação por tarefas, e acompanhamento da participação do estudante. Por fim, os serviços de coordenação focalizam o uso de bibliografias, webliografias, documentações, co-autoria de docentes e de estudantes, e *downloads* (Lucena et al., 2000).

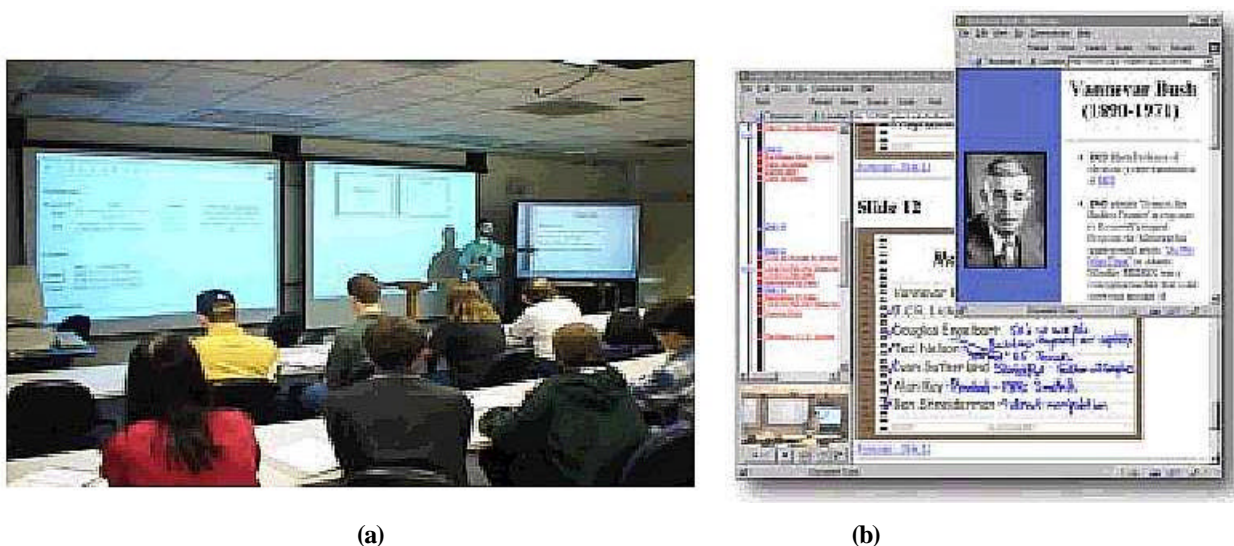
### 3.4 O ambiente eClass

O eClass, ou Classroom 2000, desenvolvido no Georgia Institute of Technology (GATECH), é um projeto que tem como principal objetivo a criação de um modo automatizado para transformar o conteúdo rico de uma sala de aula tradicional em mídias digitais extensíveis, de forma que possam ser reutilizadas a curto e longo prazo (Abowd et al., 1999)

A filosofia do eClass difere-se dos ambientes apresentados aqui por focalizar o uso do material produzido (gerado) nas salas de aula para a formação do conhecimento que, posteriormente, é disponibilizado sob a forma de hiperdocumentos aos estudantes. Segundo Pimentel (Pimentel et al., 2000), a sala de aula é um ambiente rico em recursos áudio-visuais que combinam os materiais apresentados pelo professor, as discussões em torno do tema abordado e as anotações e experiências de estudantes e professores, que, na maioria das vezes são perdidas por não se ter mecanismos adequados de captura e armazenamento dessa informação.

Do ponto de vista de *hardware*, o ambiente eClass necessita de uma sala equipada com tecnologias de computação ubíqua — lousa eletrônica, projetor, microfone, câmera de vídeo e computador em rede. Do ponto de vista de *software*, apóia-se em um conjunto de aplicações cliente-servidor — módulos para captura, sincronização, armazenamento e geração automática de hiperdocumentos baseados na infra-estrutura do ambiente WWW. Sob a visão de sistemas hipertexto, o eClass produz automaticamente, como resultado da captura de cada aula, um conjunto de hiperdocumentos multimídia (Pimentel et al., 2000). A **Figura 3.6** apresenta uma sala de aula equipada com o *hardware* citado e hiperdocumentos gerados a partir da captura da aula.

Mark Guzdial, professor e pesquisador do GATECH, projetou um ambiente hipermídia de edição colaborativa, denominado CoWeb (Guzdial, 1999). Através deste ambiente, estudantes e professores podem fazer a autoria de páginas Web sem conhecimento prévio de quaisquer tecnologias relacionadas, como comunicação cliente-servidor, linguagens de marcação e de *scripts*. Sua principal característica é a autoria aberta: qualquer usuário no ambiente CoWeb pode criar e editar as páginas, o que permite a edição colaborativa de documentos.



**Figura 3.6 – (a) Sala de aula com recursos para capturar informação da sala de aula  
(b) Apresentação da informação capturada  
(Pimentel et al., 2000)**

Observando que o ambiente CoWeb dá suporte às atividades ocorridas fora de sala de aula, enquanto que o eClass registra as atividades ocorridas dentro de sala de aula, pesquisadores do GATECH e do ICMC-USP trabalharam colaborativamente na integração desses ambientes, permitindo, assim, conectar as informações produzidas nos dois ambientes (Abowd et al., 1999), de modo que estudantes possam acrescentar suas anotações, enriquecendo ainda mais o material didático.

### **3.5 As ferramentas HyperBuilder, QuestBuilder e TaskBuilder**

No contexto de documentos didáticos estruturados, um exemplo é o projeto de um conjunto de ferramentas desenvolvido no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo – USP, do qual fazem parte as ferramentas para autoria e disponibilização de material didático no ambiente WWW. O professor utiliza as ferramentas para



criar e disponibilizar o curso, enquanto o estudante tem acesso ao material via documentos HTML e *applets* JAVA (Santos Jr., 1998), sendo que o conteúdo do material é armazenado em documentos XML.

Genericamente, o projeto consiste em três ferramentas integradas, denominadas **HyperBuilder**, **QuestBuilder** e **TaskBuilder** (Santos Jr., 1998). A ferramenta TaskBuilder se destina ao desenvolvimento de exercícios a serem inseridos no material didático, oferecendo a possibilidade de utilização de recursos multimídia como textos e imagens. A ferramenta QuestBuilder é utilizada para a criação de questionários com questões de vários tipos, tais como falso/verdadeiro, múltipla escolha e dissertativas.

A ferramenta HyperBuilder concentra todo o projeto, sendo destinada à criação e disponibilização do material didático. Nesta ferramenta, o professor pode criar um documento com o material do curso e inserir questionários e exercícios criados com a utilização das ferramentas TaskBuilder e QuestBuilder. As **Figuras 3.7, 3.8 e 3.9** apresentam as interfaces das ferramentas TaskBuilder, QuestBuilder e HyperBuilder, respectivamente.

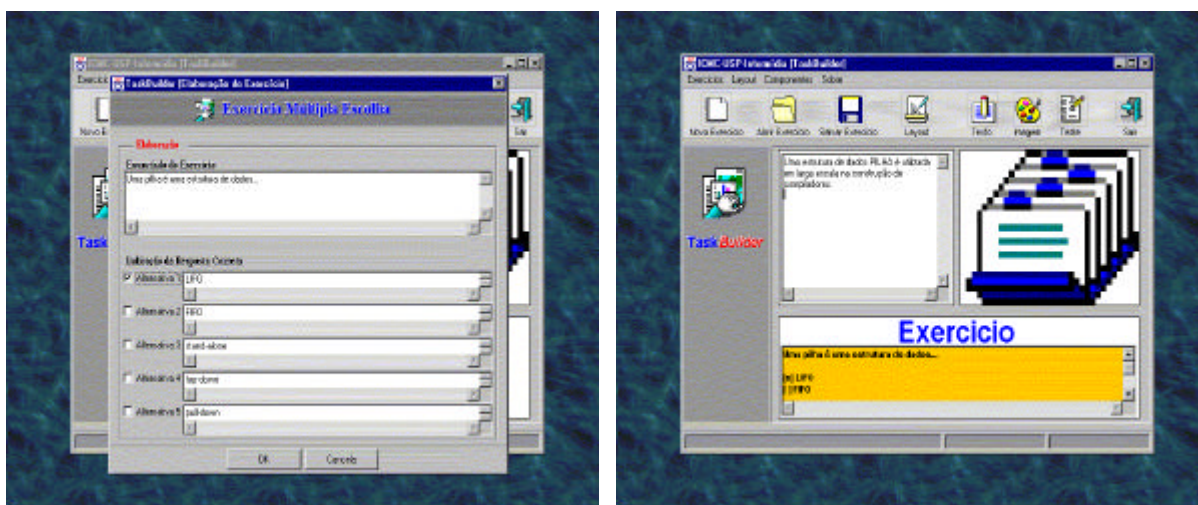


Figura 3.7 - Interfaces da ferramenta TaskBuilder (Santos Jr., 1998)

Estas ferramentas se concentram na autoria do material didático para o ambiente WWW sem considerar os mecanismos de gerenciamento do curso após a disponibilização. Desta forma, pesquisadores do Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo têm trabalhado no sentido de complementar as ferramentas de Santos Jr., por exemplo, no

armazenamento e recuperação de documentos estruturados em bases de dados (Pires & Pimentel, 2000).

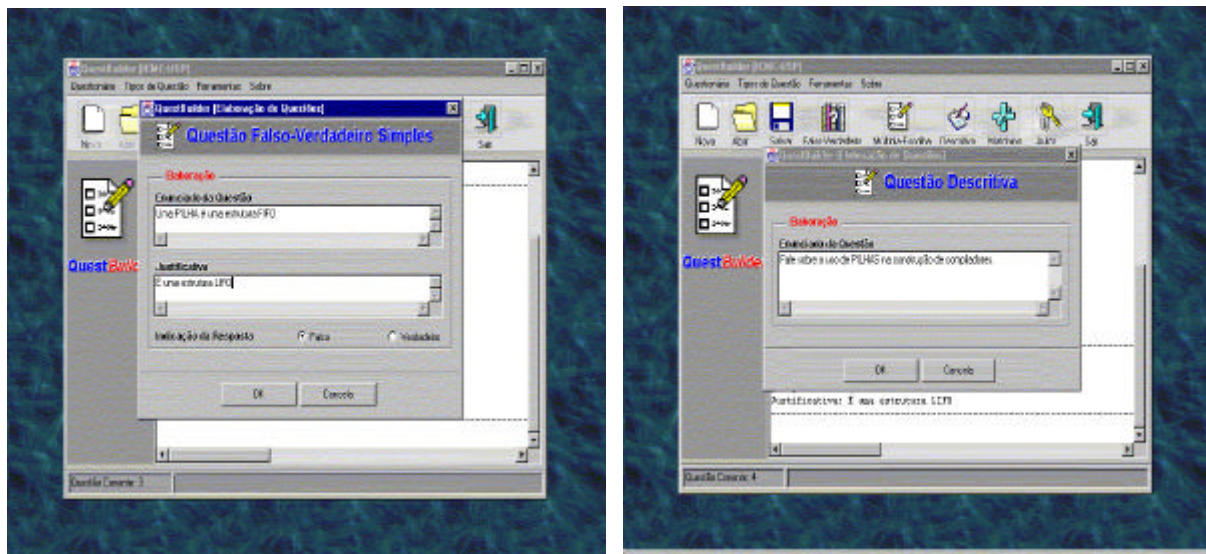


Figura 3.8 - Interfaces da ferramenta QuestBuilder (Santos Jr., 1998)

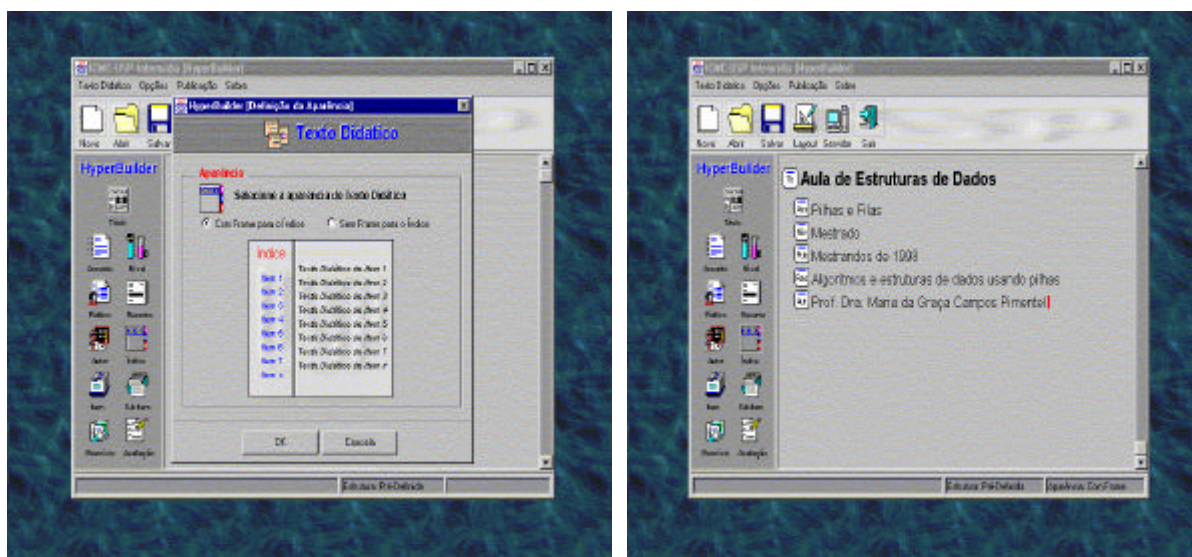


Figura 3.9 - Interfaces da ferramenta HyperBuilder (Santos Jr., 1998)

### 3.6 O ambiente *StudyConf*

O *StudyConf* é um ambiente, também desenvolvido no Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, que focaliza o uso da cooperação no ambiente WWW (Macedo, 1999). O *StudyConf* integra as ferramentas desenvolvidas por Santos Jr. para a



criação do material didático do curso, e a ferramenta *DocConf* (Pimentel et al., 1998) que permite a configuração e execução de sessões de trabalho cooperativo, oferecendo recursos de *chat*, *whiteboard* e votação.

Basicamente, o *StudyConf* considera os usuários como sendo um estudante, um professor e um administrador; e está dividido em três módulos: módulo professor, módulo administrador e módulo aluno (Macedo, 1999).

O módulo professor oferece o acesso às ferramentas para elaboração de material didático (página de *download* das ferramentas *HyperBuilder*, *QuestBuilder* e *TaskBuilder*). O módulo administrador fornece mecanismos para manutenção dos materiais disponíveis para estudo. O módulo aluno oferece suporte às atividades do estudante como o acesso à ferramenta *DocConf*.

A página principal do ambiente permite o acesso aos módulos (aluno, administrador e professor) e é apresentada na **Figura 3.10**. A página principal do módulo administrador, com informações de ajuda sobre as ações que podem ser realizadas neste módulo, pode ser visualizada na **Figura 3.11**.



**Figura 3.10 - Página principal do *StudyConf* (Macedo, 1999)**

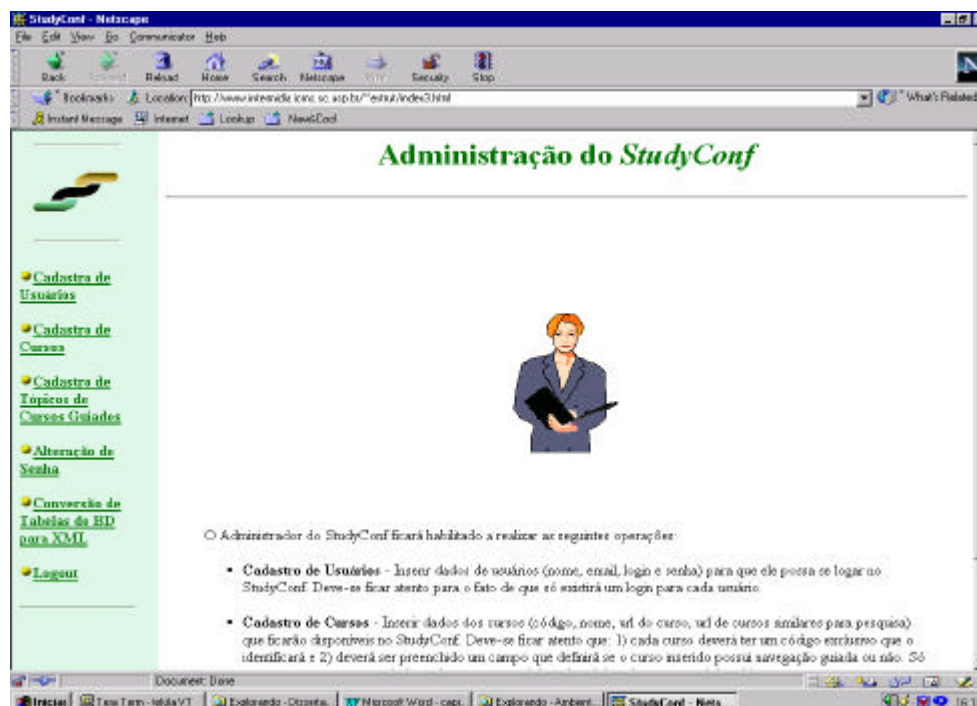


Figura 3.11 - Páginas para realização de ações nos módulos aluno e administrador (Macedo, 1999)

### 3.7 Considerações finais

Dentre os sistemas de apoio à educação, o WebCT é tido como um dos mais completos ambientes de desenvolvimento de cursos para o ambiente WWW, oferecendo uma grande variedade de ferramentas de autoria e de gerenciamento para o curso. Por outro lado, a estrutura dos cursos no WebCT é bastante proprietária, o que, de certa forma, pode dificultar o intercâmbio de informações com outras aplicações. Além do WebCT, outro ambiente que tem se destacado como bastante eficiente é o AulaNet.

Como o WebCT e o Aulanet, existem outros sistemas que não foram considerados aqui, mas que têm contribuído com a formação e desenvolvimento de sistemas educacionais cada vez melhores e mais próximos à realidade das salas de aula tradicionais.

Conforme citado, nos ambientes mais completos, existe uma certa dependência do processo de autoria com o processo de gerenciamento, o que nem sempre agrada a todos os usuários. A proposta deste trabalho é no sentido de separar as duas etapas, deixando o professor livre para utilizar o conjunto de ferramentas para geração de conteúdo que melhor se adapte ao seu contexto, e as ferramentas propostas aqui, para o gerenciamento do curso.

### 4.1 Considerações iniciais

Durante anos os sistemas computacionais foram desenvolvidos para funcionar em plataformas centralizadas onde todo o processamento de dados era realizado em um único computador. No final da década de 80, houve o crescimento da abordagem cliente/servidor, na qual as tarefas são divididas entre a máquina do usuário e um (ou mais) equipamento de apoio para operações de banco de dados, processamento de dados, dentre outras.

Na década de 90, o novo paradigma da orientação a objetos, tanto para análise, projeto e programação surgiu para facilitar o desenvolvimento de sistemas, melhorar a qualidade dos softwares e incrementar a possibilidade de reutilização de códigos. Juntamente com o paradigma de orientação a objetos, o desenvolvimento de aplicações distribuídas teve um forte impulso frente às novas tecnologias disponíveis com o advento da Internet.

Atualmente, um novo paradigma de desenvolvimento de sistemas tem se expandido com a utilização dos conceitos de agentes computacionais, que têm sido empregados no desenvolvimento de aplicações para as mais diversas áreas do conhecimento.

A tecnologia Java surge, no contexto dos agentes de software, como tecnologia base para o desenvolvimento de aplicações baseadas em agentes fixos ou móveis, devido ao seu vasto conjunto de recursos de programação distribuída.

Este capítulo apresenta uma discussão em torno da definição do termo agente, algumas propriedades encontradas nas diversas definições da literatura e uma classificação dos tipos de agentes, considerando principalmente as tarefas executadas pelos agentes. Em adição, são apresentadas as principais características da tecnologia Java, com vistas a caracterizá-la como uma linguagem adequada para o desenvolvimento de aplicações distribuídas e, conseqüentemente, de aplicações para o ambiente WWW e de agentes de software.

## 4.2 O que é um Agente?

O termo "agente" existe na linguagem dos computadores desde a década de 80, porém a definição do termo parecer ser tão complexa para a comunidade científica quanto a definição do termo "inteligência artificial".

Iniciando pela definição encontrada nos dicionários, agente é aquele que opera; ou, agente é aquele que é encarregado dos negócios de terceiros. Tais definições consideram duas perspectivas diferentes: a primeira que associa um agente a uma entidade que é capaz de agir; e a segunda em que um agente é considerado como uma ajudante, ou alguém que atua por intermédio de outra pessoa.

No contexto da Ciência da Computação, um agente de software pode ser utilizado para facilitar a criação de softwares capazes de interoperar, através da troca de informações e serviços, com outros programas, para resolver problemas complexos (Moreira & Walczowski, 1997). Porém, pode-se observar algumas outras definições feitas por pesquisadores da área:

- "Agentes de software são componentes de software capazes de comunicar e cooperar com seus pares por meio da troca de mensagens, usando uma linguagem de comunicação" (Ketchpel & Genesereth, 1994);
- "Agentes são sistemas computacionais que habitam algum ambiente complexo dinâmico, sentem e agem de maneira autônoma nesse ambiente e, assim, atingem objetivos ou cumprem tarefas para as quais foram designados" (Maes, 1994);
- "Agentes são entidades de software persistentes, dedicadas a um propósito específico. Persistência distingue agentes de subrotinas; agentes têm suas próprias idéias sobre como realizar tarefas e sua própria agenda. Propósito específico os distingue da maioria das demais aplicações; os agentes são tipicamente menores" (Jennings & Wooldrige, 1994).

Estas são algumas das definições do termo agente encontradas na literatura. Observa-se que não existe um conceito único de agente, mas conceitos diferenciados de acordo com o ponto de vista de cada um dos autores e pesquisadores. Neste ponto, vale ressaltar que a definição do termo "agente", embora seja necessária, é utilizada principalmente para classificar os sistemas computacionais e não para caracterizar, de forma absoluta, agentes e não-agentes.

### 4.3 Propriedades dos agentes

Apesar de não haver uma definição formal para o termo, é possível identificar algumas propriedades que normalmente aparecem nas definições de cada autor. Tais propriedades são relativas ao ambiente em que os agentes estão inseridos, sendo que esse ambiente constitui o meio do qual o agente obtém informação e sobre o qual o agente pode atuar. Algumas dessas propriedades são (Franklin & Graesser, 1996):

- **reatividade:** capacidade do agente de reagir ao ambiente em que está inserido e às suas mudanças, utilizando um tipo de comportamento baseado em estímulo/resposta;
- **adaptação:** capacidade do agente em adaptar-se às mudanças que ocorrem no meio em que está inserido;
- **pró-atividade:** capacidade de tomar iniciativa e não ser conduzido somente pelos eventos;
- **autonomia:** capacidade dos agentes agirem sem intervenção humana direta ou qualquer outra intervenção, tendo controle sobre o seu estado;
- **mobilidade:** capacidade que o agente tem para se mover no ambiente, inclusive de uma máquina para outra;
- **capacidade social:** capacidade de se comunicar com outros agentes, utilizando uma linguagem de comunicação de agentes;
- **continuidade temporal:** um agente não deve simplesmente executar um conjunto de tarefas e depois terminar, mas deve continuar ativo.

As várias definições de agentes permitem identificar estas e outras propriedades, o que não significa que todos os agentes tenham que, necessariamente, conter todas elas.

### 4.4 Classificação dos agentes

As propriedades dos agentes podem ser combinadas de acordo com as tarefas que executam, de modo a estabelecer condições necessárias (ou suficientes) para determinar alguns tipos de agentes. Em adição, também é possível estabelecer graus na noção de agente (Jennings & Wooldrige, 1994):

- **noção fraca de agente:** sistema computacional com as propriedades de autonomia, capacidade social, reatividade e pró-atividade;

- **noção forte de agente:** sistema computacional com as propriedades do item anterior, acrescidas de estados mentais, crenças, desejos, intenções e, algumas vezes, estados emocionais.

No contexto da Inteligência Artificial, os agentes podem ser classificados de acordo com as propriedades que apresentam, podendo ser: agentes móveis, autônomos, adaptativos, reativos e outros.

Os agentes móveis são processos com capacidades de movimentar-se pelas redes, interagindo com máquinas, coletando informações e retornando após executar os deveres determinados pelo usuário. Apesar de mobilidade não ser uma condição, nem necessária, nem suficiente para o conceito de agente, os agentes móveis apresentam algumas diferenciações sobre os estáticos, como por exemplo: redução dos custos de comunicação, computação assíncrona, arquitetura flexível para computação distribuída, além de fornecer uma abordagem atrativa e diferente para o projeto de aplicações (Chess, 1999).

Possíveis aplicações para agentes móveis incluem gerenciamento de rede, recuperação de informação, simulação distribuída, controle de dispositivos remotos, computação móvel, segurança de redes, dentre outras.

Num contexto mais amplo, quando os agentes são orientados à tarefa, é possível classificá-los com base nas tarefas que eles executam, podendo ser (Jennings & Wooldrige, 1994):

- **agentes de informação:** são sistemas capazes de obter informação requerida por um agente humano;
- **agentes de entretenimento:** são agentes que simulam e animam personalidades artificiais, normalmente em mundos virtuais destinados ao entretenimento;
- **agentes de aconselhamento:** são agentes que ajudam na execução de certas tarefas dando conselhos e sugerindo caminhos de resolução;
- **agentes assistentes:** são os que executam tarefas para o agente humano;
- **agentes de interface:** são os agentes destinados a interagir com os agentes humanos.

Existem diversas formas para implementar os agentes de software, sendo que uma delas é através da linguagem Java.

#### 4.5 A linguagem Java e suas características

A linguagem de programação Java foi desenvolvida pela Sun Microsystems, no início da década de 90, quando o objetivo principal era o uso de uma linguagem de programação que permitisse a integração total de sistemas de computação com equipamentos eletrodomésticos.

Como a linguagem Java é derivada da sintaxe da linguagem C++, os programadores familiarizados com esta linguagem podem se adaptar facilmente à linguagem Java, que estende a linguagem C++ fornecendo capacidades de sistemas distribuídos e construção de aplicações para a Internet (Thomas et al., 1997).

A linguagem Java é considerada uma tecnologia para a construção otimizada de aplicações distribuídas. Em adição, Java é multiplataforma, pois um programa escrito nesta linguagem pode ser executado em qualquer plataforma (sistema operacional) sem necessidade de alterações no código-fonte. Tal funcionalidade é possível através do processo de compilação que gera, a partir de um código-fonte, um código intermediário chamado *bytecode* que é interpretado em qualquer plataforma com suporte à Java. Devido ao vasto conjunto de recursos para programação distribuída, a linguagem Java tem sido bastante difundida na implementação dos agentes de software.

De modo geral, as principais características da linguagem Java são (Cornell & Horstman, 1997):

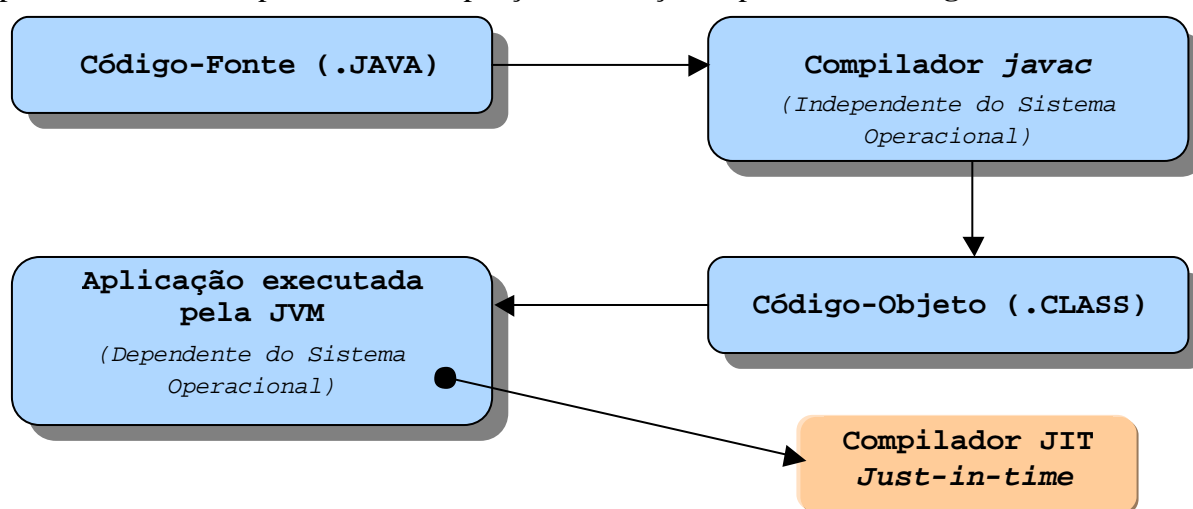
- presença de recursos *multithreaded* que permitem a implementação de aplicações multitarefa;
- através das bibliotecas AWT (*Abstract Window Toolkit*) e Swing, a linguagem apresenta um vasto conjunto de componentes para a implementação de Interfaces Gráficas com o Usuário (GUI - *Graphic User's Interface*);
- devido à sua natureza gráfica, a linguagem inclui recursos para manipulação de elementos gráficos independentemente do dispositivo gráfico e da plataforma operacional;

- possibilidade de inclusão de bibliotecas de propósito específico através da definição de interfaces denominadas *packages* (tais como bibliotecas para Inteligência Artificial, Programação Concorrente, Hipermídia, Banco de Dados e outras);
- presença de recursos para processamento e análise de imagens.

Um programa escrito em Java pode tomar dois rumos diferentes com relação à filosofia de execução do mesmo, podendo ser um *applet*, que é definido como um programa que necessita de um documento HTML e um navegador, tal como Netscape Navigator, Microsoft Internet Explorer, ou outro, para ser apresentado; ou uma aplicação *stand-alone*, que é um programa que executa diretamente em um determinado sistema operacional que promove a sua execução através de uma ordem de comando do usuário.

Apesar da explosão de uso da linguagem Java ter ocorrido com a implementação de *applets* deve-se considerar a existência de um vasto conjunto de recursos para construção de aplicações *stand-alone*.

Quando se escreve um programa em Java, esse programa recebe a extensão ".java" para identificar o código-fonte. Os compiladores Java (com ambiente de desenvolvimento ou não), possuem uma JVM (*Java Virtual Machine*) que funciona como um sistema operacional neutro. O código-fonte passa por um compilador que é responsável pela compilação do código-fonte gerando um arquivo com a extensão ".class", que é o programa-objeto gerado para esse sistema operacional neutro. O processo de compilação e execução é apresentado na **Figura 4.1**.



**Figura 4.1 - Esquema de compilação e execução de programas Java**



De posse do programa-objeto, pode-se efetuar sua execução em qualquer sistema operacional real que aceite a máquina virtual Java, tais como Windows 95, MacOS, UNIX SunOS, Linux, OS/2 Warp e outros. A JVM promove a adequação do programa objeto à arquitetura e ao sistema operacional nos quais se deseja executar o programa Java. A JVM para execução de uma aplicação *stand-alone* é dependente do sistema operacional da máquina, enquanto para a execução de um *applet*, a JVM é dependente também do *browser*. As **Figuras 4.2 e 4.3** apresentam a anatomia básica de uma aplicação *stand-alone* e de um *applet*.

```
// importação de bibliotecas
import bibliotecas_de_classes;

// abstração da aplicação
class Aplicacao
{
    public static void main (String Argumentos[])
    {
        // Instanciamento do objeto que representa o Frame
        JanelaAplicacao Janela=new JanelaAplicacao();
        Janela.show();
    }
}
class JanelaAplicacao extends Frame
{
    // Método construtor
    public JanelaAplicacao()
    {
        // código-fonte
    }
    // Método sobregravado para apresentação na interface
    public void paint(Graphics Grafico)
    {
        // código-fonte
    }
    // Método sobregravado para controle de eventos
    public boolean handleEvent(Event Evento)
    {
        // código-fonte
    }
    // Demais métodos da aplicação definidos pelo usuário
}
```

**Figura 4.2 - Anatomia básica de uma aplicação *stand-alone***

Todo *applet* Java herda um conjunto de comportamentos padrão da classe **Applet**. A execução de um *applet* obedece a um ciclo de vida de quatro estágios, cada qual possuindo um método correspondente que pode ser sobregravado (Damasceno Jr., 1996):

- inicialização (sobregavação do método *init*): este método é chamado pelo *browser* quando o objeto *applet* é carregado para execução, ocorrendo apenas uma vez no ciclo de vida;
- partida (sobregavação do método *start*): este método é chamado sempre que a página que contém o *applet* é mostrada pelo *browser*;

- parada (sobregavação do método *stop*): o *browser* faz uma chamada ao método de parada quando a página que contém o *applet* não estiver mais visível devido ao processo de navegação efetuado pelo usuário;
- destruição (sobregavação do método *destroy*): este método é chamado quando do fechamento do *browser*.

```
// importação de bibliotecas
import bibliotecas_de_classes;
// abstração do applet
class AplicacaoApplet extends Applet
{
    // Método sobregavação para inicialização
    public void init ()
    {
        // Instanciamento de objetos e variáveis
    }
    // Método sobregavação para partida do applet
    public void start()
    {
        // código-fonte
    }
    // Método sobregavação para parar a execução do applet
    public void stop()
    {
        // código-fonte
    }
    // Método sobregavação para controle de eventos
    public boolean handleEvent(Event Evento)
    {
        // código-fonte
    }
    // Demais métodos do applet definidos pelo usuário
}
```

Figura 4.3 - Anatomia básica de um *applet*

## 4.6 Desenvolvimento de aplicações distribuídas em Java

O desenvolvimento de aplicações distribuídas requer a utilização de linguagens de programação que forneçam bibliotecas de suporte à comunicação entre plataformas heterogêneas de hardware e software, e que forneçam recursos de programação que isolem os programadores das complexidades dos aspectos de comunicação, tais como padrões de conexão, detalhes de protocolos, conversões entre formatos de dados e outros. Nas subseções seguintes são apresentados alguns dos principais recursos e técnicas de programação da linguagem Java para o desenvolvimento de aplicações distribuídas, relevantes para o contexto deste trabalho.

### 4.6.1 Serviços de comunicação via URL

Os *applets* são implementados para serem anexados em documentos HTML do ambiente WWW e possuem capacidades para executar serviços de comunicação via protocolo HTTP. Um serviço

que pode ser implementado através do uso deste protocolo é a conexão a outros documentos disponíveis no ambiente WWW a partir do documento correntemente apresentado pelo *browser*, no qual o *applet* está inserido.

De modo geral, a localização das referências a outros documentos, também conhecidas como *links*, bem como dos arquivos embutidas (imagens, ícones, sons e outros) é realizada por um mecanismo denominado URL (*Uniform Resource Locator*).

A linguagem Java provê recursos para uso do mecanismo URL, permitindo que um *applet* estabeleça conexão entre o documento no qual está inserido e outros documentos que possam ser localizados de maneira uniforme no ambiente WWW (Damasceno Jr., 1996). Este recurso é extremamente útil quando se deseja, por exemplo, implementar um sistema de mapeamento das informações de um *site* no ambiente WWW. Assim, pode-se construir um mapa a partir dos nós de um documento e “navegar” pelos nós do mapa utilizando os recursos de comunicação via URL disponíveis na linguagem Java. Além desta aplicação, existem várias situações em que se torna necessário abrir documentos do ambiente WWW a partir de um *applet*.

#### **4.6.2 Aplicações *multithreading***

No princípio da computação paralela, os termos “concorrência” e “multitarefa” possuíam conceitos completamente distintos e eram aplicados a situações também distintas. Com a evolução do hardware e das técnicas de programação, estes termos passaram a ser complementares (Tanenbaum, 1996).

Entende-se por programação concorrente a implementação de técnicas que permitem a dois ou mais processos a concorrência a algum recurso comum. Neste contexto, deve-se observar que a implementação de processos concorrentes exige, dentre outros fatores, a verificação de disponibilidade do recurso comum antes de utilizá-lo. Por outro lado, entende-se por multitarefa a implementação de técnicas que permitem a execução simultânea de dois ou mais processos.

Complementando, deve-se ressaltar que técnicas de concorrência estão presentes na maioria das implementações de multitarefa, uma vez que a execução simultânea de processos pode gerar situações de concorrência (Tanenbaum, 1996).

A programação *multithreading* estende a idéia de multitarefa, fazendo com que dentro de um mesmo processo existam várias tarefas sendo executadas ao mesmo tempo, onde cada unidade computacional (subprocesso) é denominada *thread*. Quando um programa possui mais de uma unidade computacional é denominado *multithreaded*, sendo que, em termos da execução, é como se cada *thread* possuísse sua própria CPU (processador), o que, em muitos casos, pode aumentar a eficiência computacional (desempenho). De um modo geral, o uso de *threads* requer o conhecimento dos estados de execução de subprocessos. Uma *thread*, quando devidamente utilizada, passa pelos seguintes estados (Tanenbaum, 1996):

- instanciamento (*bound*): uma *thread* é definida em relação a uma classe de objetos ou outro paradigma de programação;
- criação (*fork*): uma *thread* instanciada é criada como um subprocesso que será executado simultaneamente a outros;
- partida (*start*): uma *thread* criada é colocada em estado de pronto para execução;
- execução (*run*): uma *thread* entra no estado de execução e ganha um processador;
- suspensão (*suspend*): uma *thread* é suspensa (pausa) por um determinado intervalo de tempo;
- destruição (*stop/destroy*): uma *thread* é “morta” e o processador ocupado é, então, liberado.

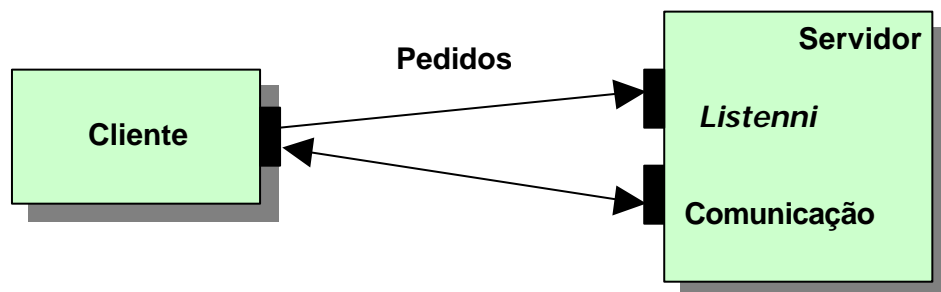
A linguagem Java possui uma classe de objetos, denominada ***Thread***, que possui métodos implementados para suportar todos os estados de execução de uma *thread*, bem como permitir o controle eficiente dos processos de sincronização e temporização (Morrison et al., 1998).

#### 4.6.3 Aplicações baseadas em comunicação via sockets

O modelo de comunicação cliente-servidor, modelo arquitetural da Internet, permite a troca de informações através de uma conexão entre as aplicações. Existem várias maneiras de se estabelecer uma conexão entre dois pontos, sendo que uma delas é a comunicação via *sockets*.

O processo para o estabelecimento de uma conexão via *sockets* inicia-se quando um cliente envia um pedido ao servidor indicando a porta (endereço lógico) para a conexão. No servidor, um módulo de software está em execução (*listenning*) e recebe o pedido do cliente; este módulo sai do estado de escuta em que permanecia, estabelece a conexão e passa ao estado de comunicação

com o cliente, no qual o servidor envia os resultados processados em relação ao pedido do cliente. O canal estabelecido pelo *socket* permanece ativo até que uma das partes, cliente ou servidor, encerre a comunicação. A **Figura 4.4** ilustra o esquema básico de funcionamento da comunicação cliente-servidor via *sockets*.



**Figura 4.4** – Esquema da comunicação cliente-servidor via *sockets*

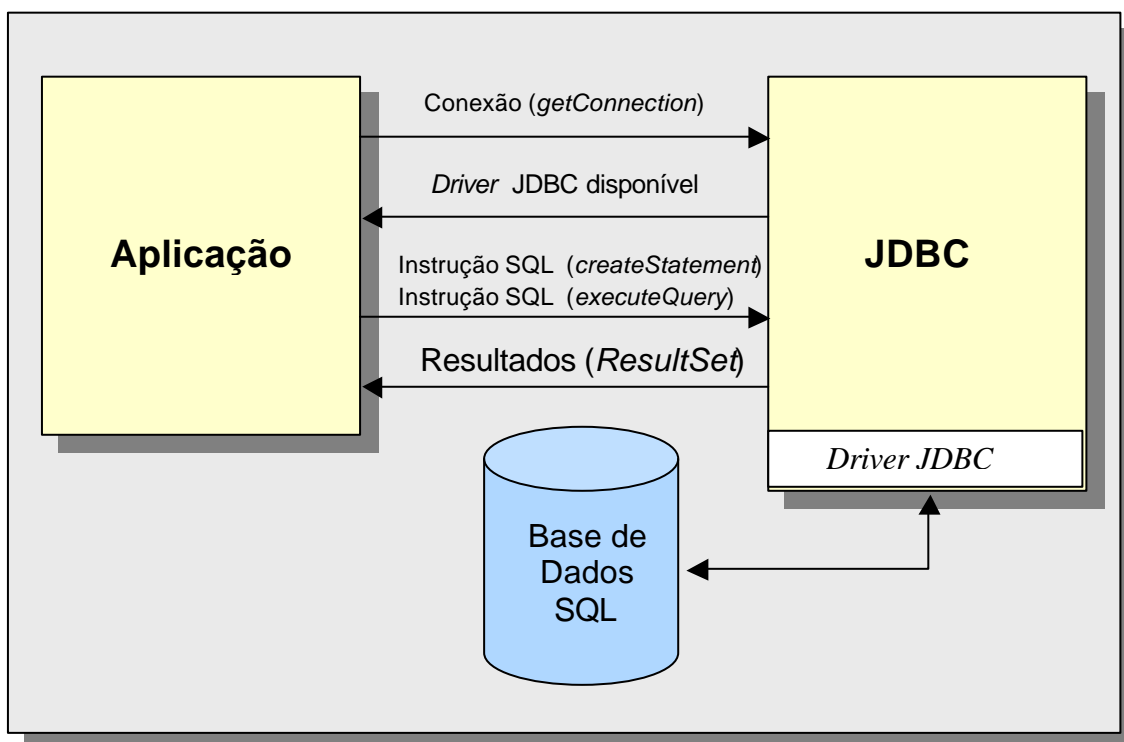
A linguagem Java oferece os recursos necessários para o estabelecimento de conexões baseadas em *sockets* entre clientes e servidores (Jamsa et al., 1997), que ocultam as complexidades do estabelecimento de uma conexão com a rede e o envio de dados por meio dela.

#### 4.6.4 Aplicações para acesso a bases de dados via JDBC

Em meados de 1995, a SUN Microsystems, tão logo foi feito o lançamento das primeiras versões da linguagem Java, passou a investigar o potencial da linguagem para o acesso a bases de dados. Neste ponto, ao invés do desenvolvimento de um gerenciador de bases de dados baseado em Java, optou-se pelo desenvolvimento de uma biblioteca de programação que permitisse o acesso a bases de dados através da linguagem SQL (*Structured Query Language*), já bastante difundida na época. Tal biblioteca, denominada JDBC (*Java DataBase Connectivity*) (Taylor, 1997), se tornou um dos pontos de maior investimento da SUN Microsystems.

A biblioteca JDBC é uma API que permite aos programadores o desenvolvimento de programas pouco complexos, em termos da codificação Java, para executar instruções SQL em ambientes distribuídos (Jepson, 1997). A combinação Java - JDBC permite o desenvolvimento de aplicações multiplataforma, visto que as bases de dados podem estar distribuídas em servidores remotos que utilizam diferentes ambientes operacionais. As versões mais novas da linguagem Java trazem esta biblioteca embutida no conjunto de bibliotecas da linguagem, não sendo mais necessária a sua instalação independente.

De modo geral, a arquitetura de software para uso do JDBC é simples. Inicialmente, uma aplicação faz um pedido ao JDBC para que seja criada uma conexão, via um *driver JDBC*, a uma base de dados. A partir do estabelecimento da conexão, um serviço de passagem de mensagens permite à aplicação o envio de *queries* (instruções SQL) ao banco de dados, que por sua vez executa os pedidos da aplicação e promove o envio dos resultados. A **Figura 4.5** ilustra a arquitetura de software utilizada para comunicação via JDBC.



**Figura 4.5 - Arquitetura do ambiente de passagem de mensagem do JDBC**

#### 4.6.5 Java Beans

Atualmente, uma característica importante para o desenvolvimento de aplicações tem sido a capacidade de reutilização de componentes, permitindo que grandes sistemas computacionais sejam projetados utilizando-se uma combinação de componentes previamente implementados e com origens possivelmente diferentes. O desenvolvimento de aplicações com base em componentes possibilita aos desenvolvedores de software obterem benefícios destas bibliotecas em diversas tarefas diferentes, principalmente em termos da economia de tempo e reutilização de código.

Através desta característica surgiu a programação visual, onde os componentes de software (tais como botões, janelas e outros) podem ser arrastados, inseridos e utilizados em novos programas. A linguagem Java expandiu um pouco mais esta forma de reutilização de componentes, inserindo o conceito de *Java Beans*. O *JavaBeans* é um conjunto de classes independente de arquitetura e de plataforma para a criação e uso de componentes de software Java (Morrison et al., 1999).

Para que os componentes sejam reutilizáveis, é necessário que eles permitam o refinamento e a reutilização em outras aplicações. A filosofia do *Java Beans* é permitir o desenvolvimento de pequenos componentes, independentes da aplicação, que possam ser facilmente manipulados por qualquer desenvolvedor. Para isto, o desenvolvimento de um *Bean* envolve alguns cuidados especiais, como por exemplo, a forma de nomear e implementar os métodos do componente, que permite a definição de mecanismos de interação entre os vários componentes.

De modo geral, um *Bean* apresenta algumas características principais que o distingue de outros objetos: propriedades, métodos, eventos, persistência e introspecção. As propriedades (ou atributos) são variáveis que afetam a aparência e o comportamento do *Bean*. Os métodos, como em qualquer outra classe, contêm códigos específicos de cada *Bean*. Um *Bean* pode se comunicar com outros *Beans* sobre algum evento ocorrido, utilizando para isto, os próprios eventos. Os eventos são mecanismos para propagar notificações de mudanças de estado entre o objeto origem e um ou mais objetos destino (*listeners*). A persistência é a característica que permite aos *Beans* salvar e restaurar seus estados. A introspecção é a característica que permite que uma aplicação detecte os métodos e variáveis disponíveis em uma classe e os reproduza sem ter acesso ao código fonte. No *Bean* a introspecção pode ser utilizada para editar suas propriedades (Morrison et al., 1999).

#### 4.6.6 Java Servlets

No contexto da arquitetura cliente-servidor da Internet, deve-se observar que o *client-side* é caracterizado pelo uso de uma interface universal de navegação (*browser*) através da qual se tem acesso aos recursos providos por um servidor. Algumas tarefas podem executadas na máquina do cliente. Por outro lado, existem tarefas que devem ser processadas pela máquina servidora (*server-side*) e os resultados enviados ao cliente.

Java *servlets* são módulos que estendem as capacidades de um servidor, permitindo o processamento otimizado de tarefas na máquina servidora. Em linhas gerais, os *servlets* são ativados por documentos HTML e as respostas são enviadas ao cliente através do protocolo HTTP (modelo de comunicação baseado em *sockets*) (Colla, 1999). Pode-se dizer que um *servlet* é semelhante ao *applet*, porém não possui interface gráfica e é executado no servidor, ao passo que um *applet* é executado na máquina do cliente.

Vale a pena observar que a tecnologia *servlet* representa um avanço em relação à tecnologia CGI (*Common Gateway Interface*). O CGI é uma interface padrão para comunicação entre o servidor HTTP e os *scripts* que executam no servidor, sendo que um *script* CGI é um programa executado independentemente, escrito em qualquer linguagem aceita pelo servidor. As linguagens mais usadas são C/C++, Perl, Shell do Unix, Tcl/Tk e Visual Basic. Um programa CGI é executado no instante em que o cliente o solicita através da sua URL (Jamsa et al., 1997).

O Java *servlet* usa uma API (*Application Programming Interface*) padrão da linguagem Java que incorpora os seus recursos de programação. O uso do *servlet* promove a otimização do que é executado no servidor, resultando em melhoras de desempenho na execução de tarefas e no envio de respostas ao cliente. Além disso, os *servlets* são inicializados apenas uma vez (quando são solicitados) ao passo que os programas CGI são inicializados e finalizados a cada pedido do cliente. A arquitetura básica de uso dos recursos dos Java *servlets* pode ser vista na **Figura 4.6**.

Quando um *servlet* aceita uma chamada da aplicação cliente, ele recebe dois objetos (Colla, 1999):

- um objeto *ServletRequest*, que encapsula a comunicação no sentido cliente-servidor;
- um objeto *ServletResponse*, que encapsula a comunicação no sentido servidor-cliente.

O uso de *servlets* permite a implementação de aplicações cliente-servidor e, no caso da Internet, particularmente do ambiente da WWW, permite a exploração dos recursos da interface universal de navegação – o *browser*. Uma maneira convencional para estabelecer um canal entre cliente e servidor é através da implementação de formulários HTML para o *client-side* e de aplicações *servlets* para o *server-side*. Também é possível estabelecer canais de comunicação entre *applets* e *servlets*.



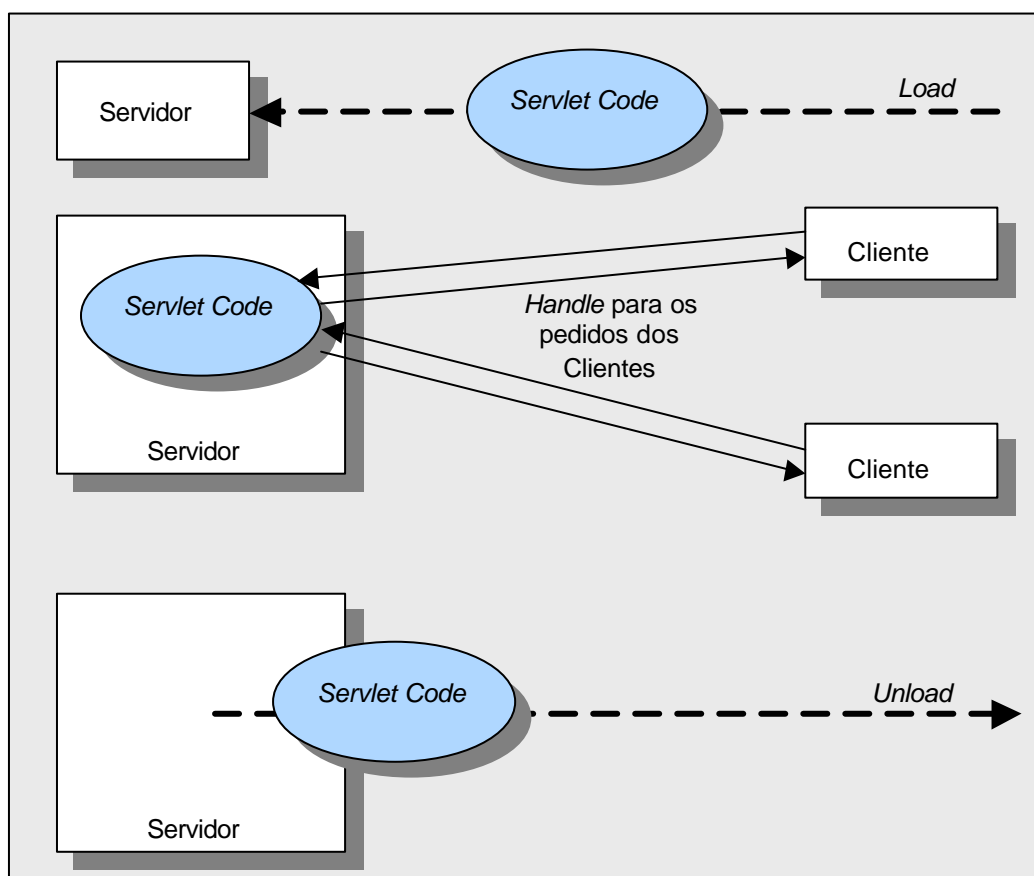


Figura 4.6 – Arquitetura *servlet* baseada em primitivas *send/receive* (Sun, 1999)

Em termos de aplicação, pode-se ressaltar algumas formas interessantes de uso para *servlets*, tais como permitir a construção de aplicações para ambientes colaborativos, gerenciamento de conferências *on-line*, processamento de requisições para ambientes de *e-commerce*, dentre outras. A capacidade de redirecionamento de pedidos, onde um *servlet* pode enviar pedidos para outros servidores ou *servlets*, também favorece o estabelecimento de políticas de balanceamento de carga entre vários servidores.

De modo geral, os *servlets* constituem uma maneira simplificada de comunicação entre a aplicação cliente e o servidor quando há a necessidade de processamento de informações neste último.

#### 4.6.7 Java Server Pages

Java Server Pages (JSP) é uma tecnologia que possibilita aos desenvolvedores e projetistas de *web* a criação rápida e de fácil manutenção de documentos. Estes documentos - páginas JSPs - são caracterizados por um documento HTML incrementado com código Java (Hall, 2000). As

páginas JSP ficam armazenadas no servidor e podem ser utilizadas para a apresentação de HTML dinâmico ou em substituição aos CGI's (*Common Gateway Interface*). A tecnologia JSP é uma extensão da tecnologia Java Servlet.

Por utilizar a linguagem Java para delimitar a lógica que gera o conteúdo do documento, as páginas JSPs podem herdar todo o conjunto de características de Java, oferecendo suporte para sistemas que exigem alta performance, escalabilidade, facilidade de manutenção e principalmente, portabilidade.

Como nas aplicações e *applets* Java, nas páginas JSPs os usuários têm a liberdade de utilizar suas próprias classes, de acordo com suas necessidades.

A compilação do código Java de uma página JSP é realizada uma única vez no servidor. Quando o usuário faz acesso a uma página JSP, se ela ainda não tiver sido compilada (acessada anteriormente) o servidor realiza a compilação do código e logo depois, a sua execução. Se a compilação já foi realizada uma vez, então somente o processo de execução é realizado. Esta característica enriquece as páginas JSP pela eficiência e facilidade de desenvolvimento, uma vez que não é necessário compilar o código manualmente a cada alteração do código Java (Hall, 2000).

Uma página JSP pode ser utilizada tanto para apresentação de informações armazenadas no servidor quanto para a obtenção e armazenamento de informações do usuário. Com as características distribuídas da linguagem Java, uma página JSP pode, por exemplo, estabelecer uma conexão com uma base de dados, através de JDBC, para recuperar ou armazenar dados. Além disso, uma página JSP pode também ser usada em conjunto com documentos XML para recuperação de informações (JavaWorld, 2000).

Em geral, as páginas JSP são utilizadas em conjunto com os formulários HTML e requerem um *Servlet Engine* (além do pacote Java) que executa no servidor de *web* (*web server*). O *Servlet Engine* é o responsável pela interpretação, compilação e execução de uma página JSP. A **Figura 4.7** apresenta a arquitetura genérica de utilização de uma JSP. A inserção do código Java no documento HTML para a criação da página JSP é feita segundo um conjunto de especificações que pode ser encontrado em (Hall, 2000).

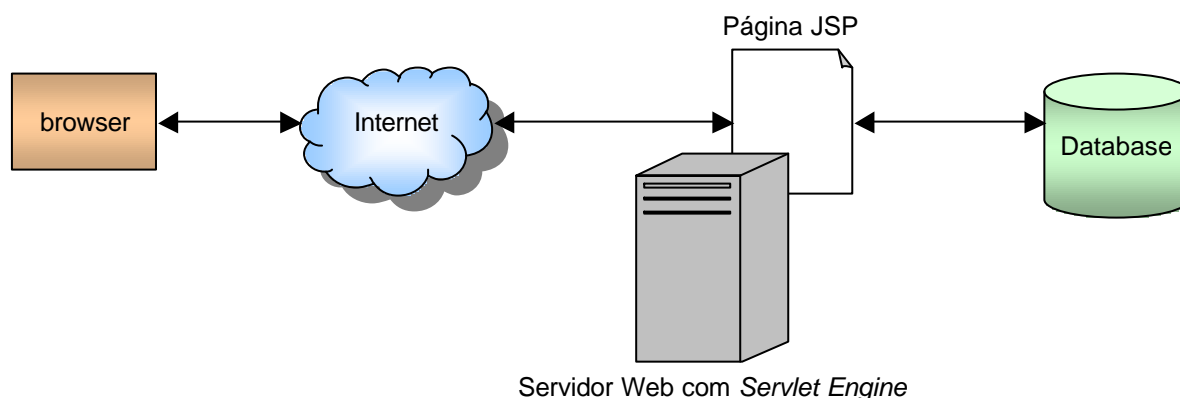


Figura 4.7 – Arquitetura genérica de utilização de JSP

A **Figura 4.8** apresenta um exemplo de um documento HTML que possui um formulário e uma página JSP. Ao submeter o formulário, a página JSP é acessada, os dados são processados e o resultado é apresentado ao usuário. Nesta figura, pode-se observar que o código Java da página JSP é exatamente o mesmo utilizado para o desenvolvimento de programas Java, sendo que a única exigência é que todo o código Java seja colocado entre as marcações `<%` e `%>`.

```

<html>
<head>
</head>

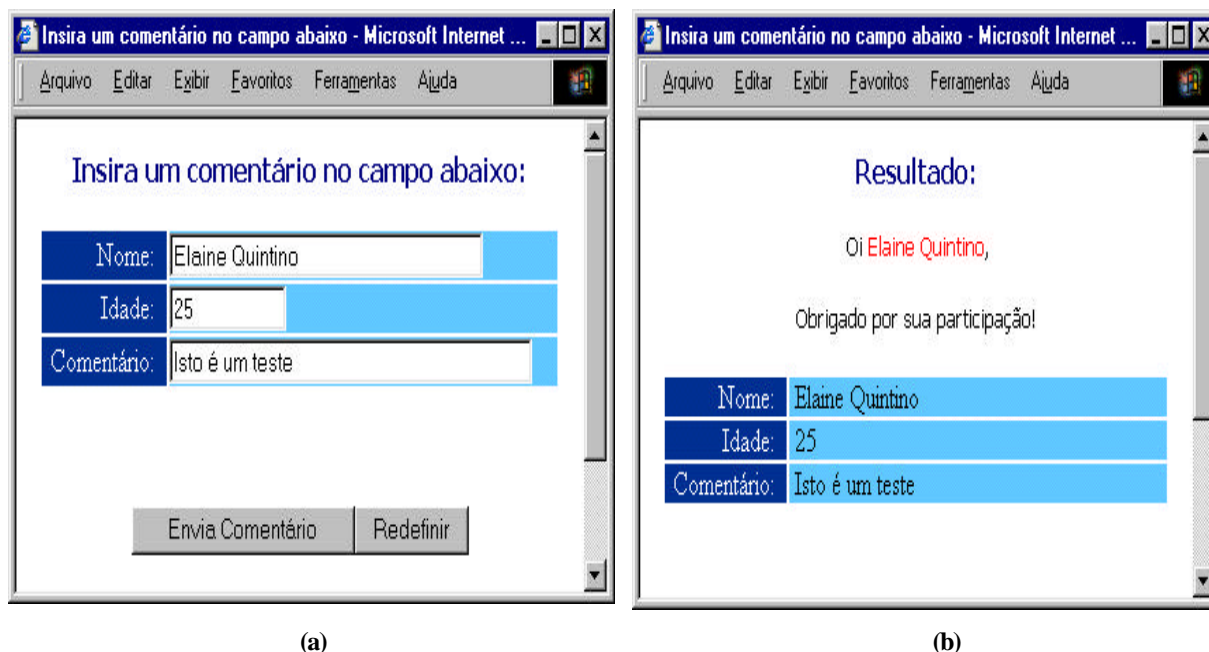
<body>
Insira um comentário no campo abaixo:
<form method="POST" action="/teste.jsp">
  Nome: <input type="text" name="nome" size="30" value="">
  Idade: <input type="text" name="idade" size="10" value="">
  Comentário: <input type="text" name="comentario" size="35" value="">
  <input type="submit" value="Envia Comentário" name="submit">
  <input type="reset" value="Redefinir" name="reset">
</form>
</body>
</html>
  
```

```

<HTML>
<HEAD>
</HEAD>
<BODY>
<%
String nome = request.getParameter("nome");
String idade = request.getParameter("idade");
String comentario = request.getParameter("comentario");
Resultado: <BR>
Oi <%=nome%>, <BR>
Obrigado por sua atenção! <BR>
<%
if (Integer.parseInt(idade) > 25) {
  Nome: <%=nome%> <BR>
  Idade: <%=idade %> <BR>
  Comentário: <%=comentario%> <BR>
}
%>
<BR>
</BODY>
</HTML>
  
```

Figura 4.8 - Exemplo de documento HTML que acessa página JSP

A página HTML do formulário no *browser* pode ser vista na **Figura 4.9a**. O resultado gerado pela execução da página JSP pode ser visto na **Figura 4.9b**.



**Figura 4.9 – (a) Documento HTML**

**(b) Resultado gerado pela página JSP**

## 4.7 Considerações Finais

O termo agente tem sido empregado de diversas formas no mundo da computação e de modo especial, no contexto da inteligência artificial. Apesar de existirem diversas definições para este termo, e não se ter chegado a um consenso em relação a este fato, as definições estão convergindo para o conceito de que algumas características fundamentais são necessárias para se caracterizar um programa como um agente. Segundo (Franklin & Graesser, 1996) as características de reatividade, autonomia, pró-atividade e continuidade temporal constituem elementos básicos para a definição de um agente.

No contexto da educação, os agentes têm sido cada vez mais explorados desde a recuperação de informação no ambiente WWW até a implementação de sistemas inteligentes de tutoria de estudantes.

Neste trabalho, o conceito de agente é inserido como assistentes que, executando um conjunto mínimo de tarefas individualmente, fornecem apoio ao estudante e ao professor diante da realização das tarefas didáticas de um curso.

Dentre as características da tecnologia Java que favorecem o desenvolvimento de agentes de software a execução em multiplataformas, que é uma característica essencial aos agentes móveis (pelo fato de serem transportados de uma máquina para outra), e o uso dos *servlets*, para a troca de informações entre o servidor e cliente. Além disso, o *Java beans* e o paradigma da orientação a objetos permitem que a linguagem Java seja utilizada para o desenvolvimento de grandes aplicações, favorecendo o reuso e a extensibilidade de seus componentes.

No contexto deste trabalho, a tecnologia Java oferece um rico conjunto de recursos para a implementação das ferramentas de gerenciamento com as características necessárias ao contexto distribuído da Internet, além dos recursos para o desenvolvimento dos agentes de software.

## 5.1 Considerações iniciais

De modo geral, pode-se dizer que o protocolo TCP/IP da Internet não oferece esquema apropriado de autenticação e criptografia dos dados envolvidos em uma conexão. As informações que trafegam pela rede através de TCP/IP são “abertas”, e qualquer pessoa com acesso ao meio físico pode acessar os dados que estiverem trafegando, bastando para isso, ter uma interface de rede em modo promísco, onde todos os pacotes são capturados mesmo que não sejam destinados a ela (Bellovins, 1989).

Para solucionar estes problemas, diversas pesquisas têm sido realizadas e diferentes soluções têm sido adotadas. Este trabalho, apresenta um novo modelo de comunicação entre clientes e servidores da Internet, utilizando-se o conceito de *views* de dados baseadas em agentes móveis. Este capítulo apresenta a definição de uma *view* de dados e como ela pode ser utilizada para o desenvolvimento de aplicações distribuídas, nas quais é necessário estabelecer conexões e transmitir dados através do protocolo TCP/IP da Internet de forma segura. As *views* de dados são utilizadas para o desenvolvimento das ferramentas de gerenciamento deste trabalho.

## 5.2 Conceituação das *Views* de dados

As *views* de dados apresentam as seguintes características:

- tecnologia padrão: a comunicação é baseada no uso do protocolo HTTP para evitar problemas com *firewalls* (barreira de proteção que impede a exploração das possíveis falhas de um sistema);
- segurança: uso de mecanismos de segurança que validam os usuários e criptografam os dados transmitidos entre o cliente e o servidor para a criação de um canal de comunicação seguro;
- facilidade de manutenção: a interface gráfica e as rotinas de manipulação de dados se encontram dentro de um mesmo objeto.

As *views* de dados podem ser definidas como pequenos agentes móveis capazes de migrar entre os computadores (cliente e servidor), encapsulando os objetos gráficos de interface e os processos de manipulação dos dados em um mesmo local (classe). A capacidade de migração dá às *views* de dados o nome de *mobile views* (*views* móveis).

Implementada na linguagem Java, a *mobile view* é composta de três partes:

- um *servlet* que pode criar *mobile views*: o *servlet Database*;
- um *applet* que pode receber e apresentar *mobile views*: o *applet Contact*;
- *mobile views* que fazem parte da aplicação (onde ficam os objetos gráficos e as rotinas de manipulação de bases de dados).

No contexto da *mobile view*, o *servlet* e o *applet* não conhecem os dados que estão sendo manipulados, o que os faz independentes da aplicação.

Para ser uma *mobile view*, um objeto deve ter um método construtor vazio e implementar a interface *View* que é descrita na **Figura 5.1**.

```
public interface View extends Serializable
{
    // Create a view - executes on server side
    public Object createView(Ticket tic, SQL db) throws Exception;

    // Initialize the GUI of the view - executes on client side
    public Panel initView();

    // Validate the view data - executes on client side
    public boolean validateView();

    // Set the new data on server - executes on server side
    public Object updateView(SQL db) throws Exception;
}
```

**Figura 5.1** – Interface *View* para implementação das *mobile views*

O acesso a uma *mobile view* é feito através do *applet Contact*. O usuário passa, nos parâmetros do documento HTML (apresentado na **Figura 5.2**), um nome de botão e a *view* que ele deseja associar àquele botão. O parâmetro *RESOURCE* é o nome da base de dados que contém as

informações a serem manipuladas. O *applet* faz a leitura dos parâmetros e apresenta a interface de acordo com os valores definidos pelo usuário. Ao pressionar um botão do *applet Contact*, um objeto da classe *ViewInterface* é instanciado para a *view* móvel associada àquele botão. A classe *ViewInterface* é responsável por fazer o pedido de autenticação do usuário e criar uma janela com um espaço para que a *view* seja apresentada ao usuário.

Embora o conceito de *mobile view* seja aplicado intensivamente no desenvolvimento das ferramentas do trabalho aqui descrito, é importante lembrar que o seu uso pode ser indicado para a implementação dos mais variados tipos de aplicações baseados no modelo de comunicação cliente/servidor da Internet, utilizando-se para isto um conjunto de classes (reutilizável) que pode ser facilmente adaptado.

```
<APPLET CODE="agents.Contact" ARCHIVE="agents.jar" width="124" height="365">
  <PARAM NAME="RESOURCE" VALUE="javaCourse"/>
  <PARAM NAME="BUTTON-0" VALUE="Info About You"/>
  <PARAM NAME="VIEW-0" VALUE="course.InfoView"/>
  <PARAM NAME="BUTTON-1" VALUE="Join Group"/>
  <PARAM NAME="VIEW-1" VALUE="course.group.groupSelectionView"/>
  . . .
</APPLET>
```

**Figura 5.2 – Passagem de parâmetros para o *applet Contact***

Para que uma *mobile view* possa ser utilizada é necessário passar por um processo de autenticação do usuário e pela criação de um canal de comunicação que seja eficiente e seguro.

### 5.3 O modelo de segurança das *views*

O processo de autenticação para acesso a uma *mobile view* é baseado no modelo de segurança do sistema Kerberos (Kerberos, 1999; Neuman & Ts'o, 1994) que utiliza *tickets* para a comunicação entre cliente e servidor.

Depois de instanciado o objeto da classe *ViewInterface* (quando o usuário pressiona um botão do *applet Contact*), é necessário criar uma conexão com o agente de usuários que está rodando no servidor (*Users Agent*) para continuar a execução da ferramenta. Este agente é responsável por criar e manipular as *views*. Antes disso, é necessário obter um *ticket* para comunicação. Neste



caso, o objeto da classe *ViewInterface* deve enviar um pedido ao agente de *tickets* do servidor (*Ticket Agent*), através do *servlet Database*, (interface de comunicação), requisitando um *ticket*. O *Ticket Agent* é responsável por verificar as informações do pedido e devolver um *ticket* à *ViewInterface*.

Quando o usuário faz *login* (utilizando a interface apresentada na **Figura 5.3** criada pela própria *ViewInterface*), o seu *username*, o nome da base de dados do curso (que é passado como parâmetro para o *applet*) e a data atual são criptografados com a senha que o usuário digita na interface. A criptografia da informação é feita utilizando-se as classes da biblioteca Cryptix (Cryptix, 1999). A data é uma informação adicional que é utilizada apenas para dificultar a “quebra” da criptografia.



Figura 5.3 – Interface de *login* para acesso às *views*

Esta informação (o *stream* gerado com as informações para solicitar o *ticket*) é enviada ao servidor para o agente *Ticket Agent*. O agente recebe o *stream* e obtém o *username* do usuário. Com este *username*, o agente obtém a senha do usuário na base de dados e tenta descriptografar a informação criptografada dentro do *stream*. A **Figura 5.4** apresenta o modelo de segurança das *views*.

Se esta operação obtiver sucesso e as informações (*username* e *resource*) forem idênticas às que vieram sem criptografia, a autenticação é válida. Então, o agente cria um objeto *Ticket* com as informações necessárias para os processos de comunicação das ferramentas, serializa (faz uma

representação do objeto em um *stream* de *bytes*), criptografa com uma senha padrão da ferramenta e envia de volta ao *applet* (à *ViewInterface*).

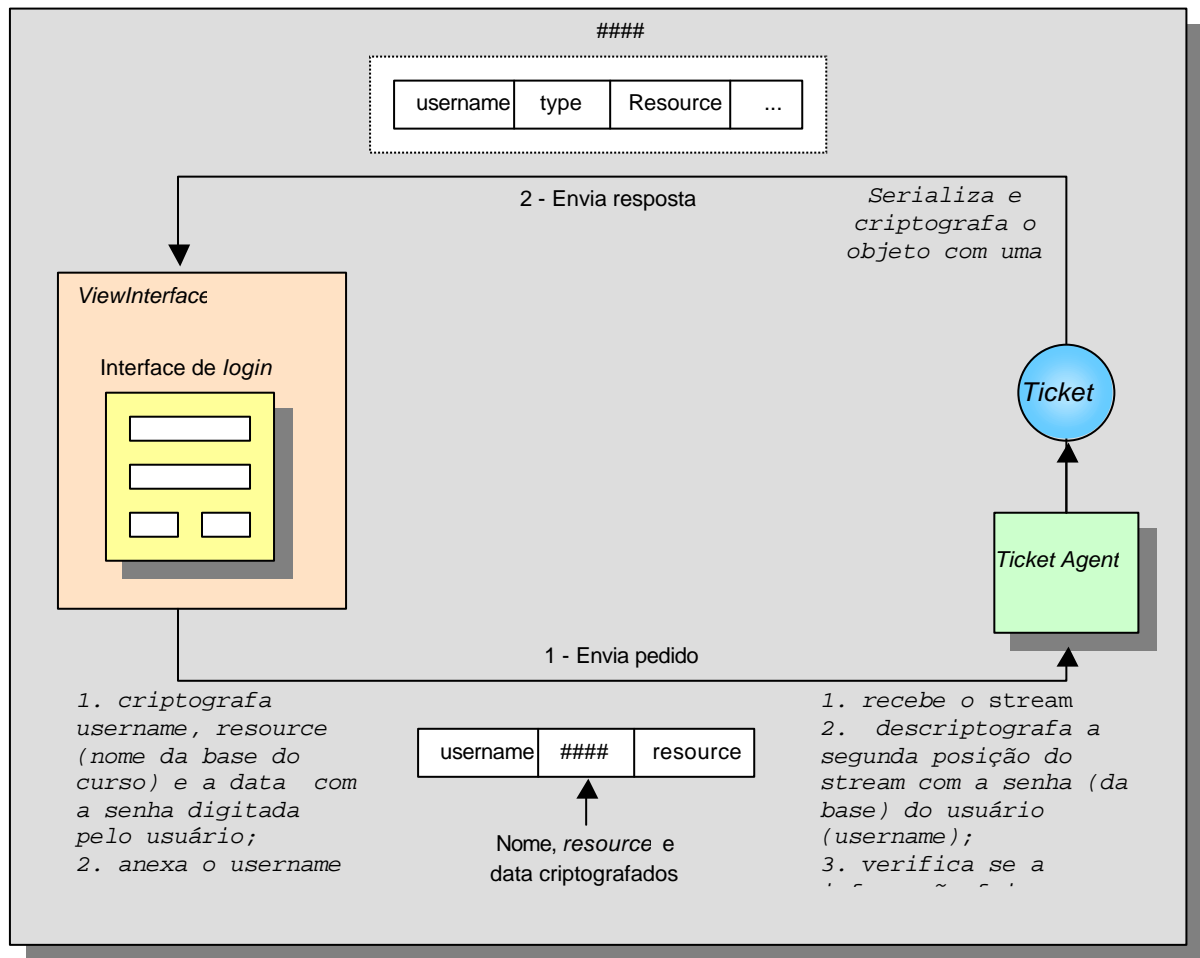


Figura 5.4 – Representação do modelo de segurança das views

De posse do *ticket*, a *ViewInterface* envia um pedido ao *Users Agent*, solicitando uma *view*. Neste caso, o pedido é enviado o nome da *view* solicitada e o *ticket* recebido do *Ticket Agent*, que permanece criptografado enquanto estiver no cliente. O *ticket* só é descriptografado quando estiver novamente no servidor. Com a criptografia da informação transmitida entre o *applet* e o *servlet*, o canal de comunicação pode ser considerado seguro.

Sempre que houver submissão de informação ao servidor, a autenticidade do *ticket* é verificada no servidor. Em adição, o *ticket* tem um prazo de validade, que, quando expirado, deixa de ser válido.

## 5.4 A mobilidade das views

O processo de movimentação da *view* inicia-se no agente *Users Agent*. A **Figura 5.5** apresenta um esquema de como as *views* podem se mover de um computador para outro.

Após receber o pedido da *ViewInterface*, o *Users Agent* descriptografa o pedido obtendo e validando o *ticket*. Logo depois, ele obtém do próprio *ticket* o nome da *view* que está sendo solicitada. Com o nome da *view*, o *Users Agent* usa o método `Class.forName("Nome da view móvel").newInstance()` para criar uma nova instância da classe que implementa esta *view*.

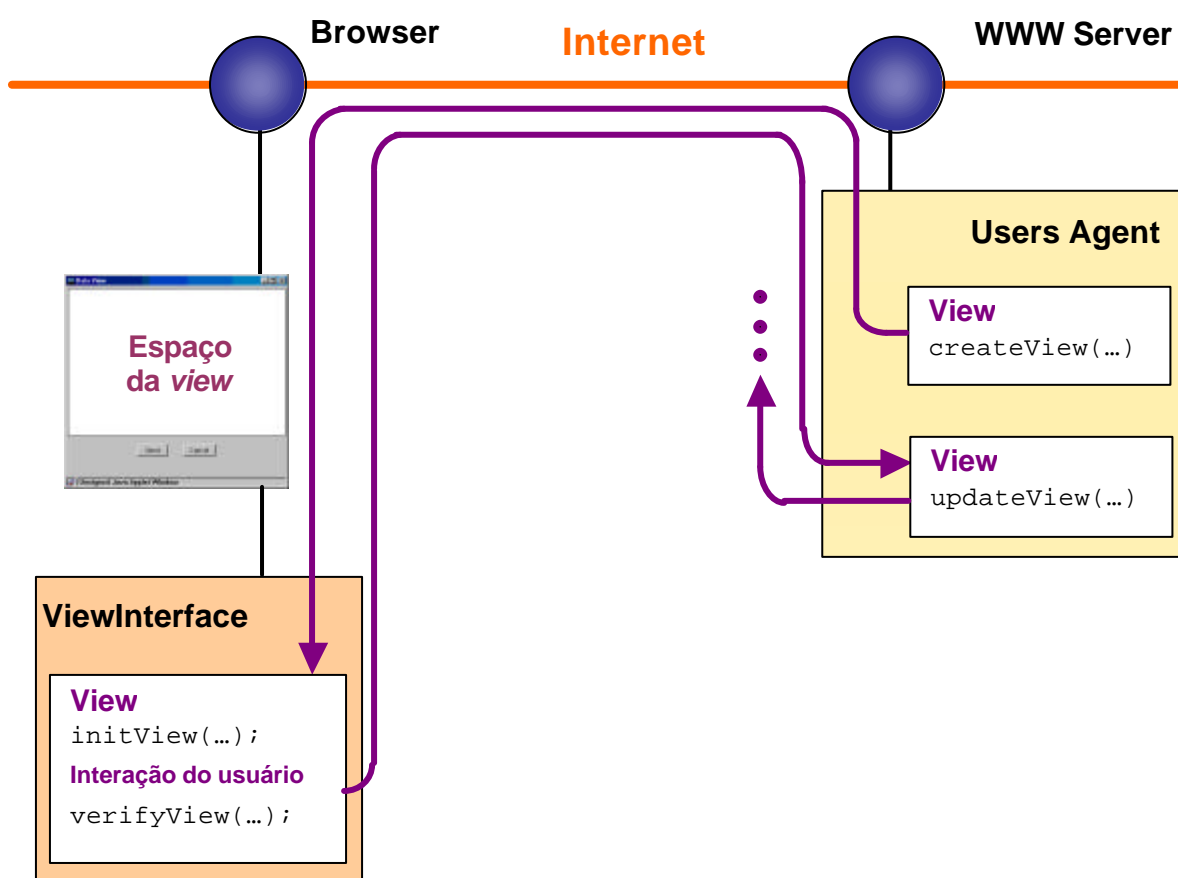


Figura 5.5 – Representação do movimento de uma *view*

Após instanciar o objeto que corresponde à *view* solicitada, o *Users Agent* faz uma chamada ao método `createView` deste novo objeto, passando como parâmetro o *ticket* descriptografado e um ponteiro (*handler*) para a base de dados. Dentro deste método, que é executado no lado servidor da aplicação, podem ser incluídos códigos de acesso à base de dados e acesso ao disco do servidor. Um exemplo pode ser a checagem do usuário para permitir ou não o acesso à *view* e

ainda, a obtenção de dados da base para manipulação. Se a execução deste método retornar com sucesso, então uma instância da *view* foi criada. O *Users Agent* serializa o objeto da *view*, criptografa e o envia de volta para a *ViewInterface*.

Na *ViewInterface* a resposta é descriptografada, o objeto (*view*) é deserializado e o método *initView* deste objeto é executado. Neste método, cria-se um painel para interface gráfica que pode conter objetos do tipo botões, caixas de seleção, menus, dentre outros objetos gráficos.

Quando o painel estiver criado, ele é adicionado em uma área reservada dentro da janela da *ViewInterface*. O painel da *view* pode ser de qualquer tamanho, pois a janela da *ViewInterface* é redimensionada automaticamente. A **Figura 5.6** apresenta uma janela da *ViewInterface* com a interface gráfica de uma *view*.

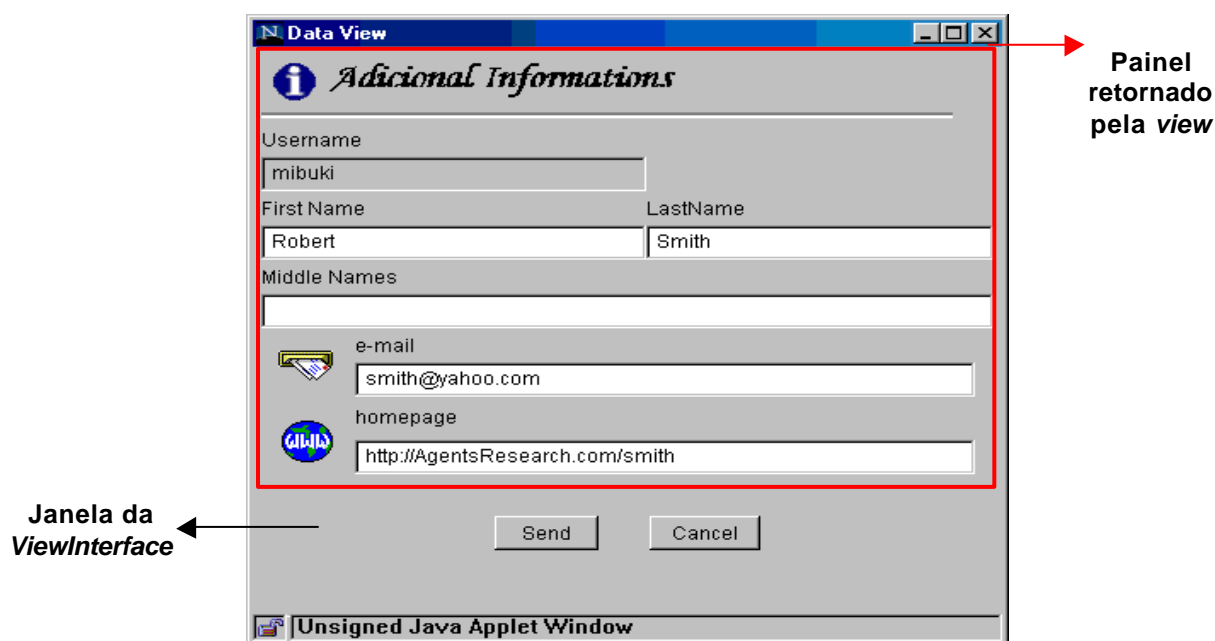


Figura 5.6 – *View* sendo apresentada na janela da *ViewInterface*

## 5.5 Interação do usuário com as *views*

Com a interface gráfica da *view* sendo apresentada, o usuário pode interagir com os objetos gráficos, inserindo textos, fazendo seleções, selecionando menus e ao final, pressionando o botão **SEND** da janela da *ViewInterface*. Quando isto ocorre, a *ViewInterface* faz uma chamada ao método *validateView* do objeto da *view*. Este método é executado no *applet* e serve para validar os dados dos objetos gráficos da *view* (por exemplo, se um e-mail tem um @). Caso este

método retorne verdadeiro, a *ViewInterface* serializa o objeto da *view*, o criptografa e envia novamente para o *Users Agent*.

No agente, o objeto é descriptografado, a *view* deserializada, e o método *updateView* executado, passando-se como parâmetros o *ticket* e o ponteiro para a base de dados (*handler*). Neste método, que executa no servidor, é possível atualizar informações na base de dados, gravar informações no disco do servidor, manipular diretórios, dentre outras operações.

Ao final da execução deste método, a *view* pode retornar dois tipos de resultados:

- uma mensagem para ser apresentada ao usuário;
- uma outra *view*.

Se a resposta for uma mensagem, a janela da *ViewInterface* é fechada e uma nova janela, com a mensagem, é apresentada ao usuário.

Se a resposta for uma outra *view*, o *Users Agent* a serializa, criptografa e envia de volta à *ViewInterface*. Novamente, o objeto é descriptografado, deserializado e o método *initView* é executado, gerando uma nova interface gráfica que é sobreposta à *view* anterior. O usuário pode interagir novamente com a nova *view*. Este ciclo continua até que a interação com o usuário termine.

## 5.6 Considerações Finais

O conceito de *mobile views* faz uso de tecnologias padrão, tais como a linguagem Java para a codificação, e o protocolo HTTP para comunicação, para oferecer uma forma eficiente e segura de transmitir informações através da Internet. Uma das principais características de uma *mobile view*, que é a possibilidade de manter a codificação da interface gráfica e as rotinas para manipulação de base de dados dentro de uma mesma classe, pode auxiliar no processo de manutenção de um sistema. Neste caso, se for necessário acrescentar ou remover informações da interface gráfica, a alteração, em termos da manipulação da base de dados, pode ser feita no mesmo local.

O modelo de segurança adotado pela *mobile view* faz com que nem a aplicação servidora (*servlet*) e nem a aplicação cliente (*applet*) saibam qual é a informação que está sendo

transmitida, ficando toda a informação concentrada na *view*. Esta característica da *mobile view* faz com que ela seja indicada para o desenvolvimento dos mais variados tipos de aplicações, variando desde grandes *sites* de comércio eletrônico até pequenas aplicações particulares.

As ferramentas deste trabalho, em sua maioria, são baseadas na utilização da *mobile views*, como pode ser observado no próximo capítulo.

### 6.1 Considerações iniciais

Conforme citado, o gerenciamento de cursos a distância via Internet pode ser visto sob dois aspectos diferentes: o gerenciamento dos materiais didáticos/conteúdos e o gerenciamento das atividades do curso. O gerenciamento dos materiais envolve aspectos de apresentados ao estudante, enquanto o gerenciamento das atividades está ligado à definição e ao controle das tarefas relacionadas ao curso, incluindo trabalhos, exercícios e avaliações.

O WebCoM - *Web Course Manager* - apresentado neste capítulo, é formado por um conjunto de ferramentas cujo objetivo é o gerenciamento, através da Internet, de um conjunto pré-definido de atividades didáticas de um curso. São apresentadas as características gerais do WebCoM, as técnicas de implementação, bem como a arquitetura e as funcionalidades das ferramentas.

### 6.2 Visão geral do WebCoM

Durante a realização de um curso, várias atividades didáticas são conduzidas com o envolvimento de diversos atores da comunidade do curso: professores (ou administradores), estudantes e monitores. Geralmente, as informações produzidas por essas atividades podem ser coletadas e disponibilizadas para uso por outros usuários interessados. Elas podem ser utilizadas para avaliação do trabalho, *feedback* para os estudantes, ponto de partida para novas atividades e outras funções. De modo geral, esse volume de informações tende a ser alto e sua manipulação, muitas vezes, pode levar a uma sobrecarga de trabalho da pessoa responsável por esta tarefa.

A proposta do WebCoM é facilitar a tarefa de gerenciamento dessas informações, de modo que os usuários possam se concentrar na parte estrutural e organizacional do gerenciamento, deixando as tarefas de baixo nível para o computador. Além disso, os usuários ainda podem contar com os recursos oferecidos pela Internet para descentralizar o gerenciamento das atividades.

Depois de analisar e avaliar os procedimentos mais comuns em um ambiente de educação, um conjunto de atividades de gerenciamento foi selecionado como base para a implementação do WebCoM:

- **Assignments:** são tipos de atividades que permitem a realização de trabalhos em grupo de estudantes. Cada *assignment* é composto por um ou vários projetos que podem ser atividades de pesquisas bibliográficas, trabalhos de desenvolvimento, realização de experiências ou outras atividades, dependendo do tipo de curso que está sendo gerenciado. Cada projeto de um *assignment* pode ser feito por um ou mais grupos. Os resultados obtidos podem ser disponibilizados para acesso por outros interessados. Para este tipo de atividade, o professor tem ainda a possibilidade de estimular a opinião crítica dos estudantes utilizando para isto o mecanismo de revisão (*reviews*). Neste mecanismo, os trabalhos são trocados entre os estudantes cada grupo deve avaliar os resultados do outro. Depois disso, uma seção de debate pode ser realizada para que o grupo revisor aponte os problemas encontrados nos resultados do grupo revisado, enquanto este último se defende e apresenta novas soluções;
- **Reports:** são trabalhos realizados individualmente que também devem gerar relatórios (que podem ser disponibilizados para outros interessados). Estas atividades podem ser as mesmas desenvolvidas na atividade *assignment*, porém sem a formação de grupos e sem os mecanismos de revisões. Os *reports* também podem ser avaliações que o estudante recebe para serem feitas em casa e, posteriormente, entregues ao professor.
- **Tests:** envolvem as atividades individuais, por exemplo, avaliações ou exercícios, que normalmente não são disponibilizadas para outros interessados, mas que devem ter notas associadas a elas e precisam ser consideradas no gerenciamento.

As ferramentas do WebCoM foram projetadas para o controle destas atividades, sendo que o processo de gerenciamento inicia-se com a criação de um ambiente, por um usuário administrador, de acordo com o plano de atividades do curso. Depois de criado e configurado o ambiente, as ferramentas devem ser disponibilizadas no ambiente WWW. As funcionalidades do WebCoM são descritas mais adiante.



De modo geral, as seguintes características podem ser observadas no WebCoM:

- independência de plataforma: implementado na linguagem Java, o WebCoM herda suas características de execução em múltiplas plataformas;
- independência do ambiente de autoria e do material didático: o acesso às ferramentas é feito através de *applets* que são independentes de qualquer ambiente de autoria;
- possibilidade de escolha na utilização dos recursos das ferramentas: através do recurso de passagem de parâmetros para o *applet* é possível definir quais recursos das ferramentas utilizar;
- controle de acesso baseado no tipo de usuário: o botão de acesso às ferramentas do estudante e do professor é visível por todos os usuários, mas o acesso às ferramentas do professor é controlado com base no tipo de usuário que faz o *login*;
- facilidade de utilização do conjunto de ferramentas: basta inserir o *applet* em um documento HTML e configurar o lado servidor para permitir a utilização de todos os recursos oferecidos pelas ferramentas;
- gerenciamento feito através do ambiente WWW da Internet: os *applets* podem ser executados em qualquer *browser* que possua uma JVM;
- independência de recursos extra na máquina cliente: por utilizar o conjunto componentes padrão da biblioteca AWT, não necessita de *plug-ins* no *browser* do cliente.

Com base nas características e no conjunto de atividades apresentados, o WebCoM é proposto com vista a suprir a carência de ferramentas específicas para gerenciamento de cursos na Internet, permitindo o gerenciamento eficiente da informação produzida em um curso. O WebCoM focaliza o controle, obtenção, armazenamento e divulgação desta informação de forma simplificada e distribuída, utilizando o ambiente WWW e outros recursos da Internet.

### 6.3 Arquitetura do WebCoM

O WebCoM foi projetado segundo os conceitos de agentes apresentados no **Capítulo 4**. No contexto deste trabalho, o conceito de agentes assistentes é o que melhor se adapta às aplicações propostas. Os agentes assistentes deste trabalho visam fornecer apoio ao estudante e ao professor diante da realização das tarefas de um curso. A **Figura 6.1** apresenta a arquitetura genérica do WebCoM. Os agentes são responsáveis pela realização das tarefas de baixo nível do gerenciamento (como acesso a bases de dados e manipulação de espaços físicos em disco),

enquanto estudantes e professores utilizam-se de interfaces gráficas para as tarefas de mais alto nível (como entrada de informações solicitadas pelos agentes).

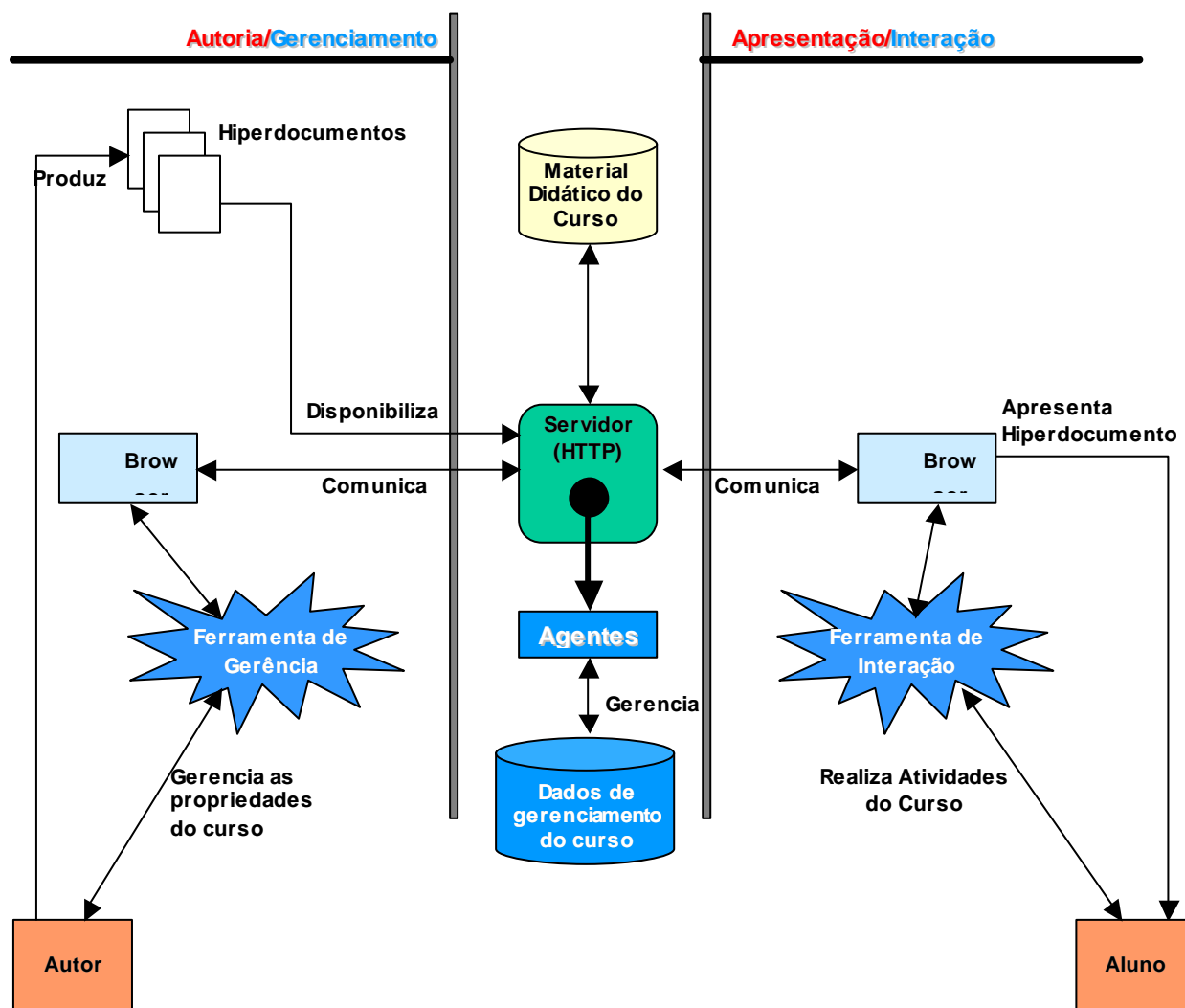
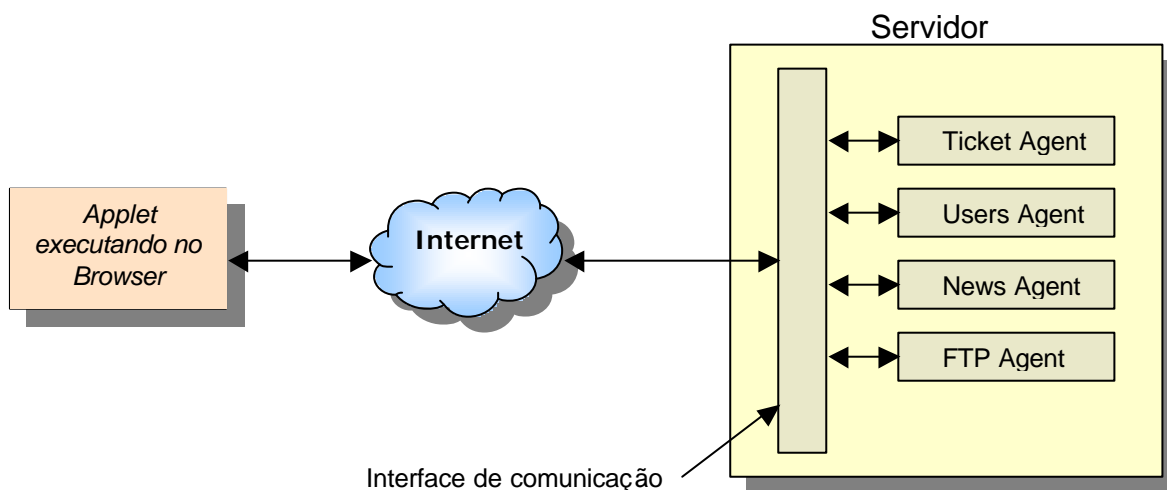


Figura 6.1 – Arquitetura geral do WebCoM

Pela **Figura 6.1** pode-se observar que a interação com as ferramentas do WebCoM é feita através do ambiente WWW (um *browser* comum) com ferramentas específicas para os professores (e/ou administradores) e para os estudantes, cujo acesso é controlado pelo processo de autenticação. A informação gerenciada nas ferramentas do professor forma a base para as ferramentas do estudante e são utilizadas para controlar as suas ações (por exemplo, verificar se a data para entrega do trabalho não expirou).

Os agentes assistentes envolvidos neste trabalho não são dotados de inteligência, mas utilizam o princípio de cooperação, onde cada agente executa uma função e, cooperando, completam uma

tarefa específica e complexa. O WebCoM é formado por um grupo de quatro agentes que executam no lado do servidor e que se comunicam com o *applet* do lado do cliente através de uma interface de comunicação. A arquitetura de agentes do WebCoM é representada na **Figura 6.2**.



**Figura 6.2 – Arquitetura dos agentes do WebCoM**

O agente *Ticket Agent* recebe os pedidos de autenticação da interface de *login* e processa o pedido retornando um *ticket*, conforme apresentado no **Capítulo 5**, caso a autenticação tenha sido válida ou, uma mensagem de erro, caso algum problema tenha ocorrido durante a autenticação.

O agente *Users Agent* é responsável pelo tratamento das *views* da maioria das ferramentas, conforme apresentado no **Capítulo 5**.

O agente *FTP Agent* é utilizado para realizar as operações de transferência de arquivos do cliente para o servidor, através de um *applet* assinado.

O agente *News Agent* é utilizado para enviar *e-mails* aos participantes do *Newsgroup*. Ele recebe a mensagem escrita em um formulário HTML e a envia a todos os estudantes do curso.

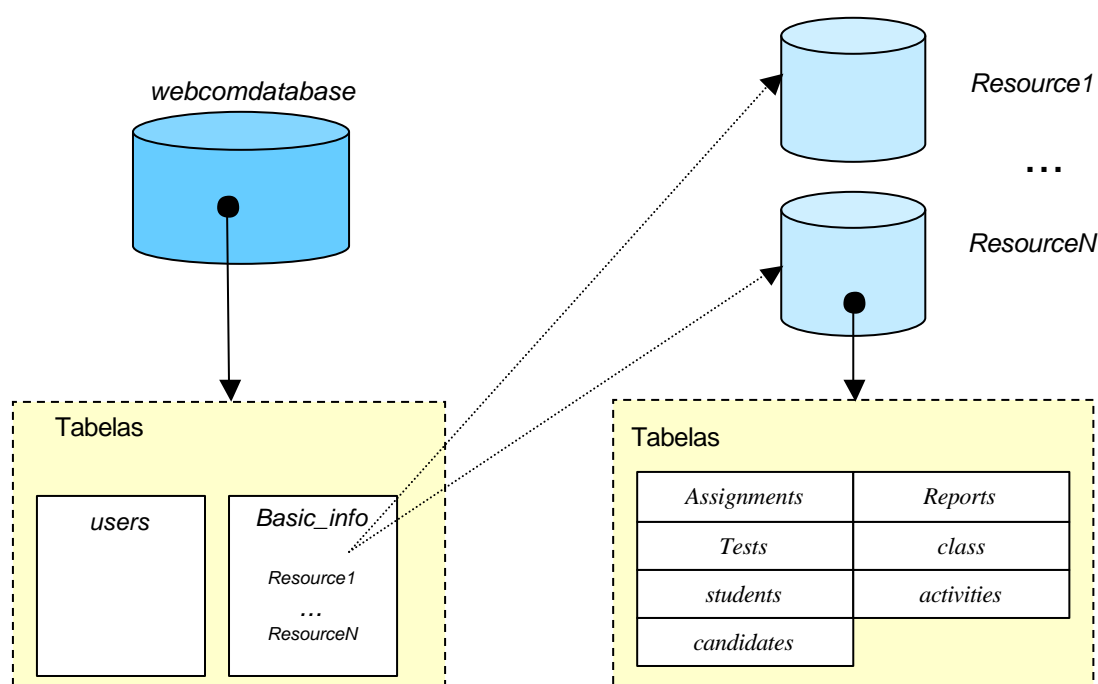
Além dos agentes, as páginas JSP também fazem parte da arquitetura do WebCoM, sendo que o seu acesso pode ou não requerer a autenticação do usuário. Neste trabalho, as páginas JSP são utilizadas tanto para apresentação quanto para recuperação e obtenção de informações.

A interface de comunicação do *applet* com os agentes é um *servlet* Java genérico que recebe o pedido do *applet* e o direciona para o agente solicitado, mantendo um canal de comunicação (baseado em *socket*) aberto para receber os resultados do agente e enviar de volta ao *applet*. Este *servlet* faz parte de um pacote para desenvolvimento das chamadas *views* de dados, que são descritas na próxima seção.

#### 6.4 Armazenamento de dados no WebCoM

Os dados coletados nas interfaces gráficas do WebCoM são armazenados em base de dados relacionais para servirem de base para novas operações. Nas ferramentas, a consulta a estes dados é feita via *queries* SQL.

O WebCoM conta com uma base de dados para armazenar as informações comuns aos cursos, e com uma base de dados específica para cada curso gerenciado, como pode ser observado na **Figura 6.3**.



**Figura 6.3 – Organização das informações do WebCoM**

A base *webcomdatabase* é utilizada tanto pelo *Ticket Agent* para validar os usuários no momento de *login*, quanto pelas *views*, para obter os dados específicos de cada curso que está sendo

gerenciado pelo WebCoM. A outra base, *basic\_info*, armazena, especificamente, as informações associadas a um curso, tais como estudantes de cada turma, atividades associadas a cada estudante, notas dos estudantes, informações gerais sobre as atividades do curso. As tabelas que formam as bases de dados são descritas na **Figura 6.4**.

```
users = {username, password, type, last_name, first_name, middle_names, email, homepage,
creation, question, answer}
```

```
basic_info = {database_name, course_name, course_date, directory, html_directory,
grade_step, average}
```

```
class = {id, expire, creation}
grade_step = {name, limit_grade}
students = {username, id, class}
monitors = {username, directory, expire}
candidates = {accepted, username, city, state, address, ...}
reports = {id, date, weight, class}
tests = {id, date, weight, class}
assignments = {id, date, review_date, weight, use_review, class}
projects = {username, type, id, group_name, grade, class}
activities = {username, type, id, group_name, grade, class}
groups = {name, assignment, project, reviews, grade, review_grade, creation}
```

**Figura 6.4 – Organização das tabelas**

## 6.5 Acesso ao WebCoM

Conforme citado, por ser implementado com recursos padrão da biblioteca AWT da linguagem Java, o uso do WebCoM depende apenas da configuração do servidor que irá hospedá-lo.

Para utilizar o WebCoM em uma página do ambiente WWW, o usuário precisa ter, em um servidor que esteja conectado à internet, os seguintes componentes:

- um módulo de servidor para execução de *servlets*;
- um módulo de servidor para execução de páginas JSP;
- um Sistema Gerenciador de banco de dados (SGBD);

É importante ressaltar que os diretórios de *servlets* e de JSP (definidos na instalação dos módulos) devem ser especificados e protegidos, de modo que *servlets* e páginas JSP somente possam ser executadas em um único local. Isto evita, por exemplo, que um estudante faça a

postagem de um arquivo JSP que altere ou acesse informações do servidor sem autorização prévia.

A configuração do WebCoM é simples, uma vez que todos os arquivos necessários para a sua execução estão centralizados dentro do diretório `manager_files`, onde ficam os seguintes arquivos:

- Database.class: *servlet* que corresponde à interface de comunicação com os agentes;
- UploadServlet.class: *servlet* utilizado para fazer *upload* de arquivos via protocolo HTTP para o servidor;
- Server.jar: arquivo compactado que contém os agentes;
- Agents.jar: arquivo compactado que contém as classes que fazem parte do processo de comunicação cliente/servidor (por exemplo, *ViewInterface*, *TicketInterface*, dentre outras);
- FtpApplet.jar: *applet* assinado para transferência de arquivos;
- um conjunto de arquivos (“*.jsp*”) que corresponde às páginas JSP;
- alguns arquivos de imagem (“*.gif*”) utilizados nas páginas JSP.

Inicialmente, este diretório deve ser copiado para dentro de um diretório do servidor. Os *servlets* que estão neste diretório devem ser movidos para o diretório de *servlets* (configurado quando da instalação do módulo servidor de *servlets*), juntamente com o arquivo Server.jar. Este último deve ficar em um diretório protegido por conter informações-chave de criptografia.

Na página HTML que dará acesso às ferramentas do WebCoM, é necessário incluir o *applet* passando o caminho dos arquivos Agents.jar e FtpApplet.jar. No diretório `manager_files`, existe uma página HTML, apresentada na **Figura 6.5**, que apresenta as instruções para a utilização do WebCoM.

Embora na **seção 5.4** tenha sido apresentado que o *applet Contact* aceita parâmetros para definição do nome dos botões e das *views*, na implementação do WebCoM um novo *applet* foi implementado e customizado, principalmente pela necessidade de utilização de recursos adicionais, tais como as páginas JSP.

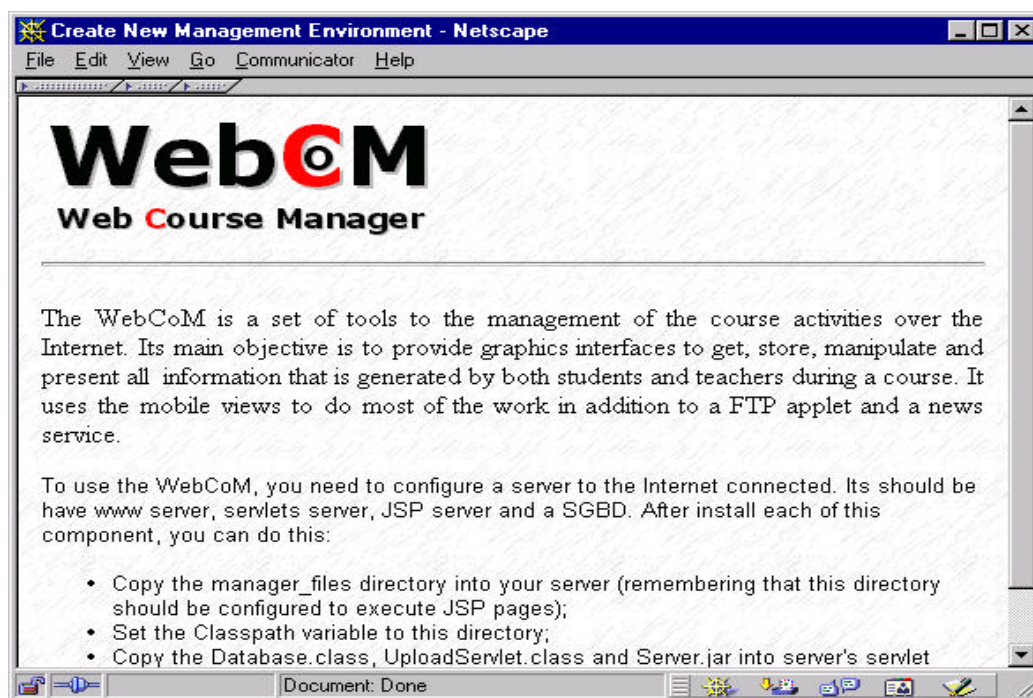


Figura 6.5 – Página principal de acesso ao WebCoM

## 6.6 As Funcionalidades do WebCoM

Com o servidor configurado, o WebCoM pode ser, então, utilizado. A **Figura 6.6** apresenta as funções de cada tipo de usuário: administrador, monitor e estudante. O controle sobre o acesso a cada ferramenta é feito através do tipo de usuário que faz o *login*, sendo que a sua validação acontece dentro da *view* no método `createView`.

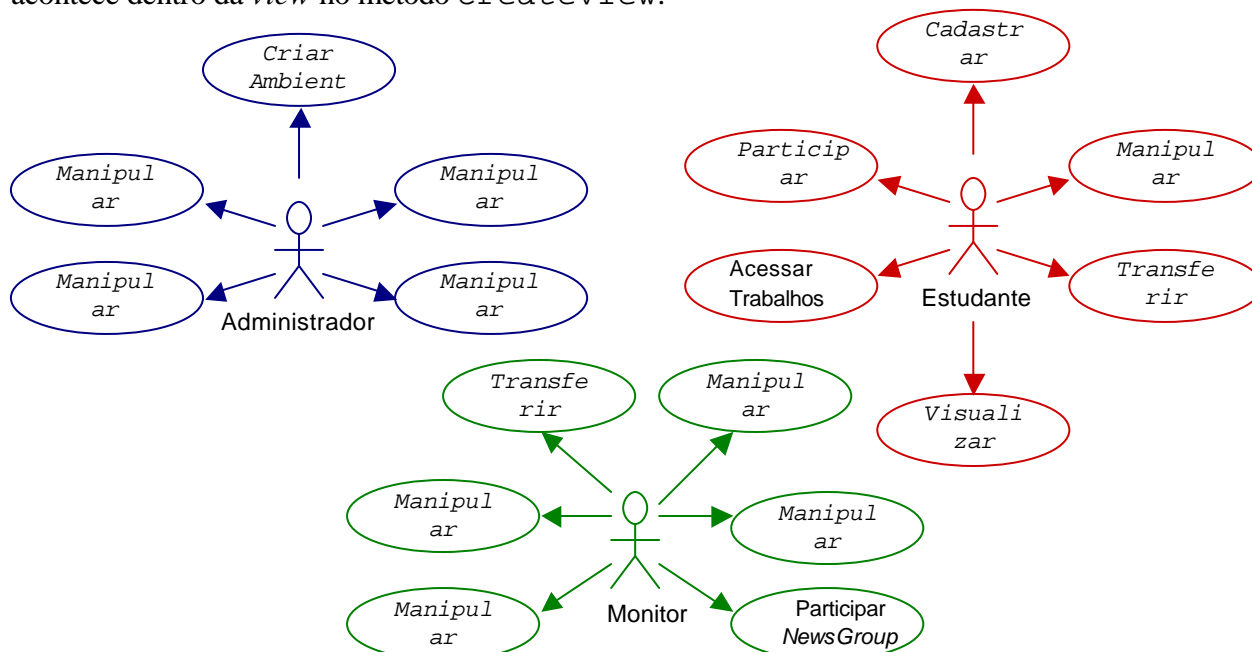


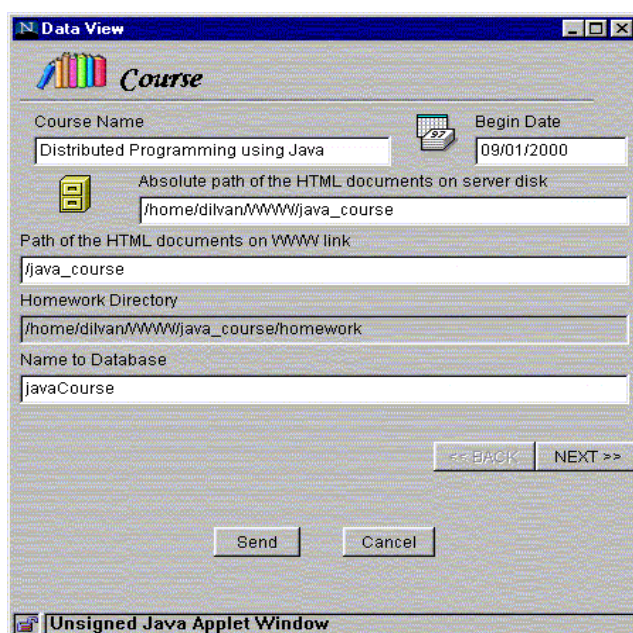
Figura 6.6 – Funções de cada usuário do WebCoM



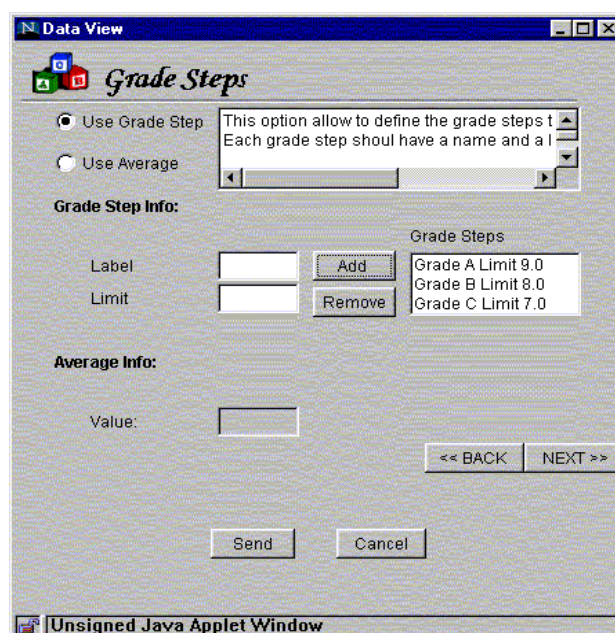
### 6.6.1 A criação do ambiente

A utilização do WebCoM deve começar com a criação de um ambiente de gerenciamento. O acesso a esta ferramenta é feito através de uma página – NewCourse.html – que também fica no diretório `manager_files`. Neste processo, o usuário administrador utiliza uma interface gráfica para definir a base do diretório *homeworks*, o nome da base de dados, a data de início do curso e a forma de aprovação dos estudantes (médias aritméticas ou conceitos com médias ponderadas).

Na criação do ambiente, o subdiretório *homeworks* é criado para a alocação dos arquivos dos estudantes. Este diretório contém outros subdiretórios, que são: *assignments*, *reviews*, *reports* e *trash*. Nos subdiretórios *assignments* e *reviews* ficam armazenados os trabalhos feitos por grupos de estudantes, enquanto no diretório *reports* ficam os trabalhos individuais. No subdiretório *trash* são armazenados os diretórios removidos dos outros subdiretórios (remoção feita através da ferramenta). As **Figuras 6.7a** e **6.7b** apresentam interfaces do processo de criação do ambiente.



(a)



(b)

**Figura 6.7 – (a) - Definição das informações básicas do curso**

**(b) - Definição da forma de avaliação do estudante (notas)**



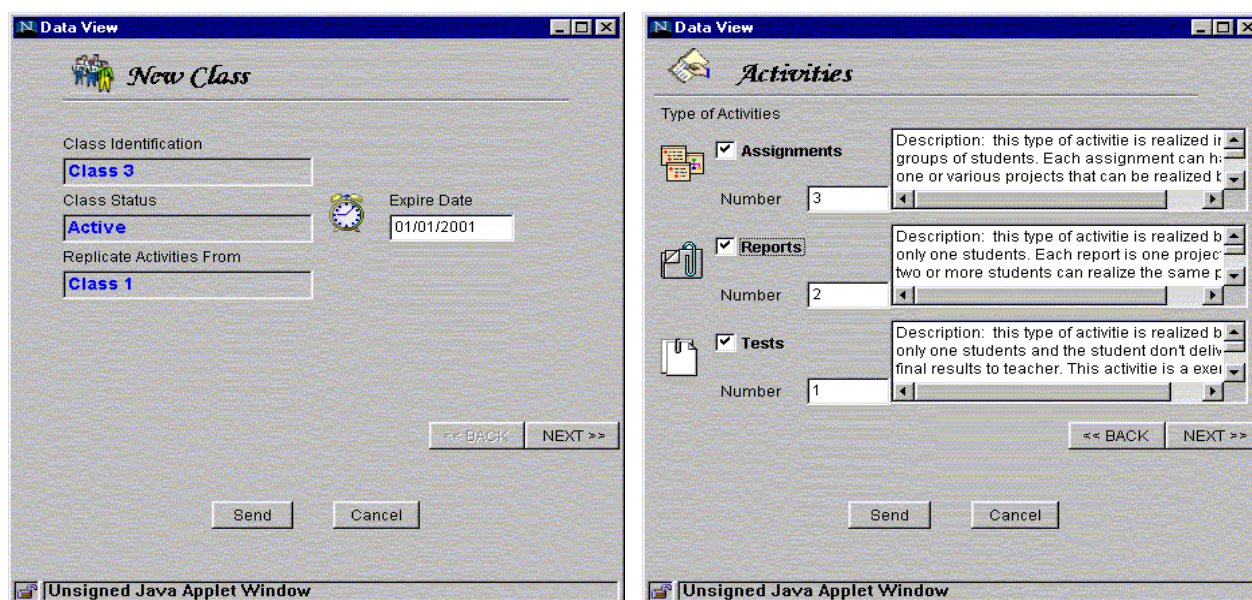
Além da criação do diretório *homeworks*, ao criar um ambiente, as informações gerais do curso são armazenadas na tabela *basic\_info* (da base *webcomdatabase*) e uma nova base de dados é criada para armazenar as informações específicas do curso que está sendo criado.

Somente depois de criar o ambiente, o usuário administrador tem a opção de criar novos administradores, monitores ou turmas de estudantes. O cadastro de estudantes depende da existência das turmas.

### 6.6.2 O gerenciamento das turmas de estudantes

A criação das turmas envolve a definição da data em que os estudantes deixam de ter acesso às ferramentas e a definição do número e tipo de atividades previstas para o curso (*assignment*, *report* e *test*), bem como sua especificação em termos de projetos e datas de entrega.

Se o curso já possuir outra turma, o administrador tem a possibilidade de copiar as informações da outra turma para a nova, ou editar as informações da turma existente. As **Figuras 6.8a** e **6.8b** apresentam as interfaces para a definição dos dados da turma e para a escolha das atividades do curso.



The figure consists of two side-by-side screenshots of a Java applet window titled 'Data View'.  
Screenshot (a) is titled 'New Class'. It contains the following fields:

- 'Class Identification' with a text box containing 'Class 3'.
- 'Class Status' with a dropdown menu showing 'Active'.
- 'Replicate Activities From' with a dropdown menu showing 'Class 1'.
- 'Expire Date' with a date picker showing '01/01/2001'.

At the bottom, there are buttons for '<< BACK', 'NEXT >>', 'Send', and 'Cancel'.  
Screenshot (b) is titled 'Activities'. It contains the following fields:

- 'Type of Activities' with three checked checkboxes: 'Assignments', 'Reports', and 'Tests'. Each checkbox has a corresponding 'Number' field and a text area for a description.
  - 'Assignments': Number '3', description: 'Description: this type of activitie is realized in groups of students. Each assignment can h: one or various projects that can be realized t'.
  - 'Reports': Number '2', description: 'Description: this type of activitie is realized b only one students. Each report is one projec two or more students can realize the same p'.
  - 'Tests': Number '1', description: 'Description: this type of activitie is realized b only one students and the student don't deliv final results to teacher. This activitie is a exer'.

At the bottom, there are buttons for '<< BACK', 'NEXT >>', 'Send', and 'Cancel'.

(a)

(b)

Figura 6.8 – (a) - Definição dos dados da turma

(b) - Definição do tipo e número de atividades

As **Figuras 6.9a** e **6.9b** apresentam interfaces para a definição de uma atividade *assignment* e de uma atividade *report* (a interface da atividade *test* é semelhante à da atividade *report*). A possibilidade da existência de várias turmas permite a realização do mesmo curso para mais de um grupo de estudantes com datas diferentes para entrega de trabalhos.

**(a) Assignments**

Assignment: 1 of 3  
 DeadLine Date: 10/08/2000  
☒ Use Review  
 Review Date: 10/10/2000  
 Weight (1-9): 1

Projects to Assignment 0

Name Project	List of Projects
	Project String_Changer MaxGroup 8 MinStudents 3 MaxStudents 3
	Project Lisp_Interpreter MaxGroup 8 MinStudents 3 MaxStudents 3
	Project ULEncoder MaxGroup 8 MinStudents 3 MaxStudents 3
	Project Internal_Number MaxGroup 8 MinStudents 3 MaxStudents 3

Number Max of Groups: 1  
 Min Students for Group: 1  
 Max Students for Group: 1

Buttons: Add, Remove, << BACK, NEXT >>, Send, Cancel

Unsigned Java Applet Window

**(b) Reports**

Report: 1 of 1

DeadLine Date: 09/09/2000  
 Weight (1-9): 2

Buttons: << BACK, NEXT >>, Send, Cancel

Unsigned Java Applet Window

(a)

(b)

**Figura 6.9 – (a) - Definição de uma atividade *assignment***

**(b) - Definição de uma atividade *report***

Estas interfaces são apresentadas em sequência, de modo que o processo só termina na interface final apresentada ao usuário.

Uma característica importante que deve ser ressaltada é que as ferramentas não estão ligadas ao material didático do curso. Assim, a definição de cada atividade didática deve ser feita nos hiperdocumentos que formam o material do curso. As informações sobre as atividades inseridas nas interfaces do WebCoM ficam armazenadas e servem como base para as ferramentas de interação do estudante e do professor (monitor ou administrador).

### 6.6.3 O gerenciamento dos usuários

Com o ambiente e as turmas criadas, os usuários podem ser cadastrados (com exceção dos usuários administrador e monitor que não dependem da existência de uma turma). O cadastramento de um usuário administrador deve ser feito por outro usuário administrador, e do



mesmo modo com o usuário monitor. As **Figuras 6.10a** e **6.10b** apresentam as interfaces para cadastrar administradores e monitores.

A informação *Access Directory* na interface do cadastro de monitor corresponde ao diretório do curso que o usuário terá acesso para fazer transferência de arquivos (o *default* é o diretório base para os *homeworks*).

The figure consists of two side-by-side screenshots of Java applet windows. Both windows have a title bar that says 'Data View' and a status bar that says 'Unsigned Java Applet Window'.

Window (a) is titled 'Administrator' and contains the following fields:

- Username: admin1
- First Name: Administrador
- Last Name: Primeiro
- Type Password: \*\*\*\*\*
- Retype Password: \*\*\*\*\*
- E-mail: admin1@domain.com

There are 'Send' and 'Cancel' buttons at the bottom.

Window (b) is titled 'Monitor' and contains the following fields:

- Username: monitor
- First Name: Jose
- Last Name: Silva
- Type Password: \*\*\*\*\*
- Retype Password: \*\*\*\*\*
- Access Directory (ex.: /teste or /teste/homework): /teste
- E-mail: jose@xxx.com.br
- Expire Date: 12/30/2000

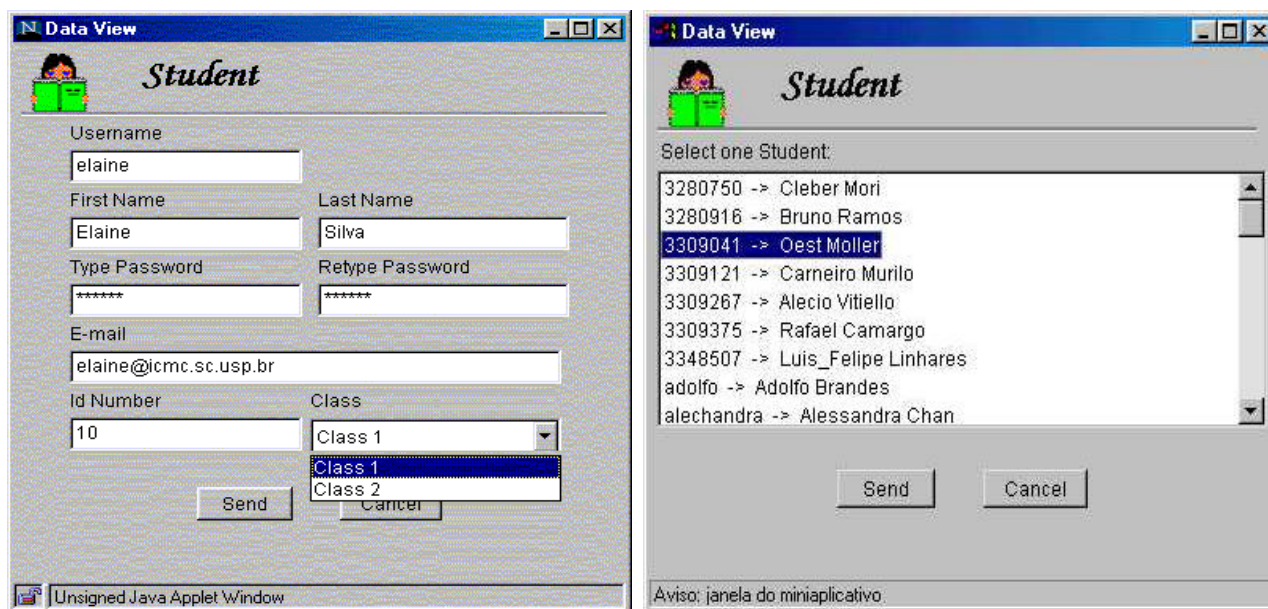
There are 'Send' and 'Cancel' buttons at the bottom.

(a) (b)  
**Figura 6.10 – (a) Interface para cadastro de administrador para o curso**  
**(b) Interface para cadastro de monitores para o curso**

O usuário estudante pode ser cadastrado tanto por um administrador quanto por um monitor e depende da existência da turma, por ser diretamente ligado a ela. Além de adicionados, os três tipos de usuários podem ser editados ou removidos. As **Figuras 6.11a** e **6.11b** apresentam duas interfaces, sendo que a primeira é a interface para cadastrar um estudante, e a segunda, uma interface para a escolha de um estudante a ser removido do curso.

O WebCoM utiliza o serviço de e-mail da Internet, de modo que se um usuário (seja administrador, monitor ou estudante) é cadastrado, removido ou tem suas informações alteradas, ele recebe por e-mail o resultado da operação. Se ele for cadastrado, recebe informações para acesso às ferramentas; se for removido, recebe notificação de exclusão; e se tiver suas informações alteradas, recebe as novas informações.

Além do cadastro de usuários, o WebCoM conta com um sistema automático de cadastro que permite que os estudantes da Internet interessados no curso façam um pré-cadastro (candidate-se ao curso). Neste sistema, os interessados têm acesso a um formulário HTML (página JSP) para inserir as suas informações, incluindo a sugestão de um *username*.



(a) (b)  
 Figura 6.11 – (a) Interface para cadastro de estudante do curso  
 (b) Interface para escolha de estudante a ser removido do curso

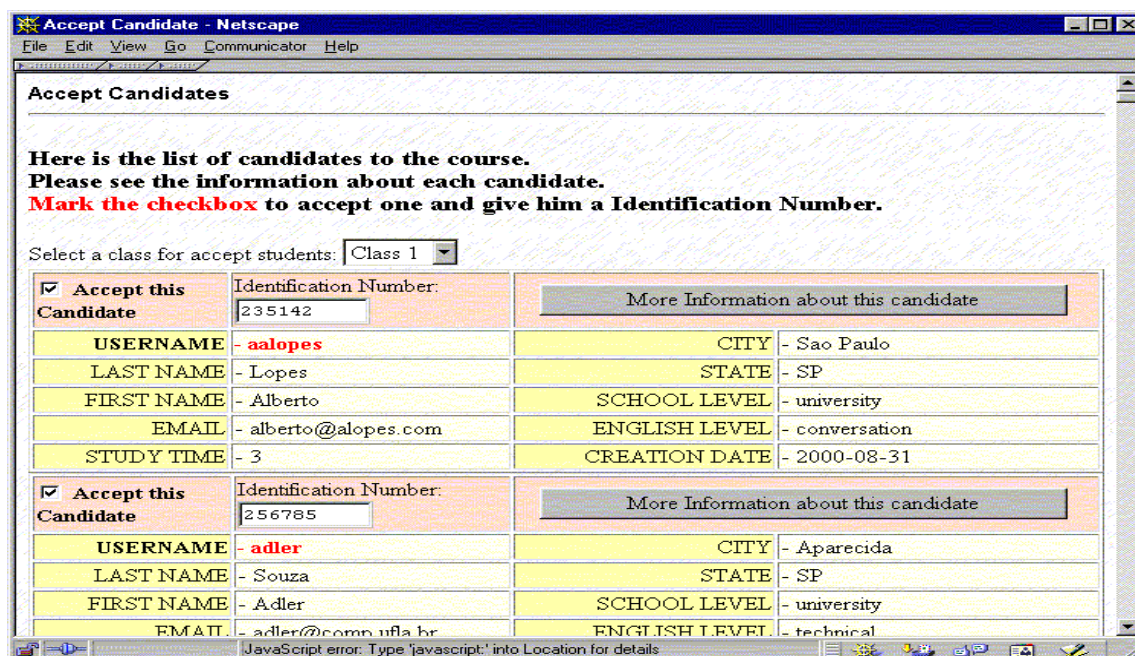


Figura 6.12 – Interface para aceitação de candidatos



Do lado do administrador (ou monitor), existe uma ferramenta que permite fazer a aceitação e cadastro automático dos candidatos. Nesta ferramenta, todos os interessados são listados num formulário e o administrador tem a possibilidade de marcar os candidatos selecionados e, ainda, adicionar um número de identificação que pode ser utilizado posteriormente (como um número de matrícula, por exemplo). A **Figuras 6.12** apresenta a interface para a aceitação dos candidatos.

#### 6.6.4 O gerenciamento das atividades didáticas

Com a área de gerenciamento criada e com os usuários cadastrados, o curso pode começar. Todos os usuários cadastrados podem acessar as ferramentas para alterar suas informações pessoais (tais como nomes, e-mail, *homepage*) ou para alterar suas senhas, como pode ser visto nas **Figuras 6.13a** e **6.13b**.

Como um dos principais objetivos do WebCoM é gerenciar as atividades didáticas obtendo e disponibilizando informações, uma das principais ações dos estudantes é a postagem dos trabalhos desenvolvidos durante o curso. Neste ponto, torna-se importante ressaltar que a forma de distribuição do material didático e o controle do estudante no uso desse material não são tratados no contexto deste trabalho.

The figure consists of two side-by-side screenshots of Java applet windows. Both windows have a title bar that says 'Data View'.  
Screenshot (a) is titled 'Adicional Informations' and contains several text input fields: 'Username' (filled with 'vanderley'), 'First Name' (filled with 'Vanderley'), 'LastName' (filled with 'Rosa'), 'Middle Names' (empty), 'e-mail' (filled with 'vrosoa@icmc.sc.usp.br'), and 'homepage' (empty). There are 'Send' and 'Cancel' buttons at the bottom.  
Screenshot (b) is titled 'Password' and contains: 'Username' (filled with 'vanderley'), 'New Password' (filled with asterisks), 'Retype New Password' (filled with asterisks), a 'Question to remember password' (filled with 'qual o nome do meu cachorro?'), and an 'Answer to the question' (filled with 'bob'). There are 'Send' and 'Cancel' buttons at the bottom.  
Both windows have a status bar at the bottom that says 'Unsigned Java Applet Window'.

(a) Interface para visualização e alteração dos dados pessoais  
(b) Interface para alteração de senhas de usuários

De acordo com as atividades definidas, existem duas formas de postar arquivos: a entrega dos *reports* e a entrega dos *assignments* e *reviews*.

Para a entrega do *report*, cada estudante cadastrado no curso possui um diretório onde os arquivos postados por ele devem ficar armazenados. Quando o estudante faz *login* para a transferência de arquivos, a ferramenta identifica o seu diretório automaticamente, de modo que o usuário não pode navegar nos níveis superiores.

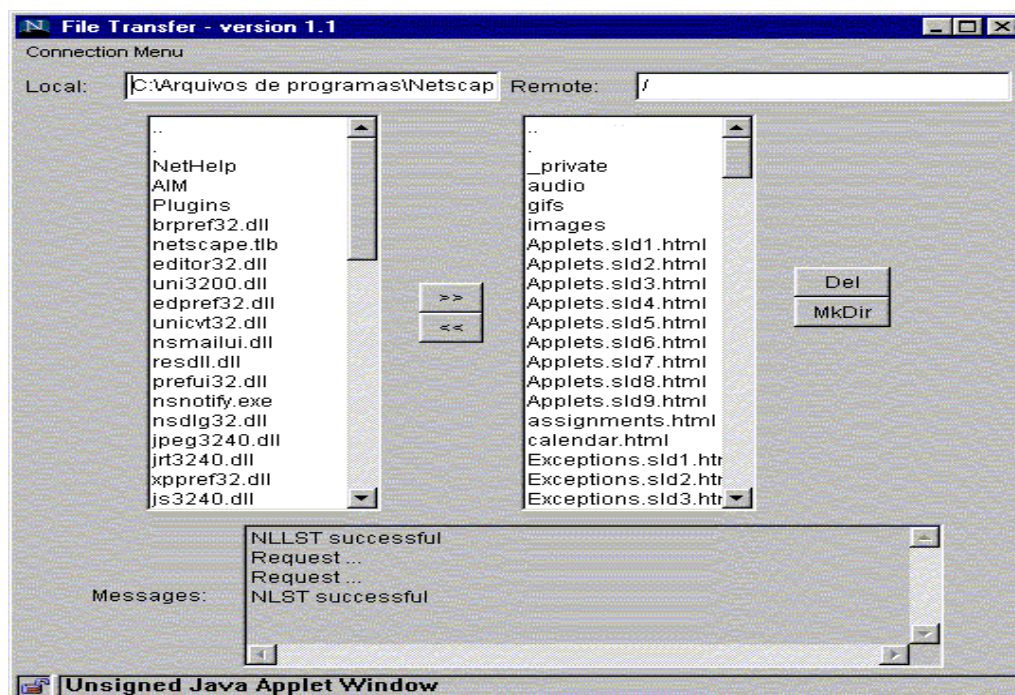


Figura 6.14 – Interface para transferência de arquivos via *applet*

A **Figura 6.14** apresenta a interface do *applet FTP* assinado para a transferência de arquivos dos usuários. A **Figura 6.15** apresenta a interface da ferramenta para fazer *upload* de arquivos via formulários HTML. Esta forma de transferência de arquivos pode ser menos problemática por não ser necessário obter certificados para a sua utilização e por ser baseada no protocolo HTTP.

Para a entrega do *assignment*, os estudantes devem se cadastrar em um grupo de trabalho. A ferramenta para a formação dos grupos tem a capacidade de controlar um número mínimo e máximo de estudantes por grupo e o número de grupos que podem fazer um determinado projeto de um *assignment*. Além disso, a ferramenta verifica se o estudante não faz parte de outro grupo ou se os integrantes não são de turmas diferentes. As informações que oferecem suporte a esta ferramenta são provenientes da definição das atividades no momento da criação das turmas.

Cada grupo também possui um diretório para armazenar seus arquivos. Quando um integrante de um grupo faz acesso à ferramenta de transferência de arquivos, a ferramenta identifica o grupo do estudante e direciona os arquivos para o diretório do grupo. O grupo pode acessar a ferramenta para transferir arquivos de *assignments* ou *reviews*.

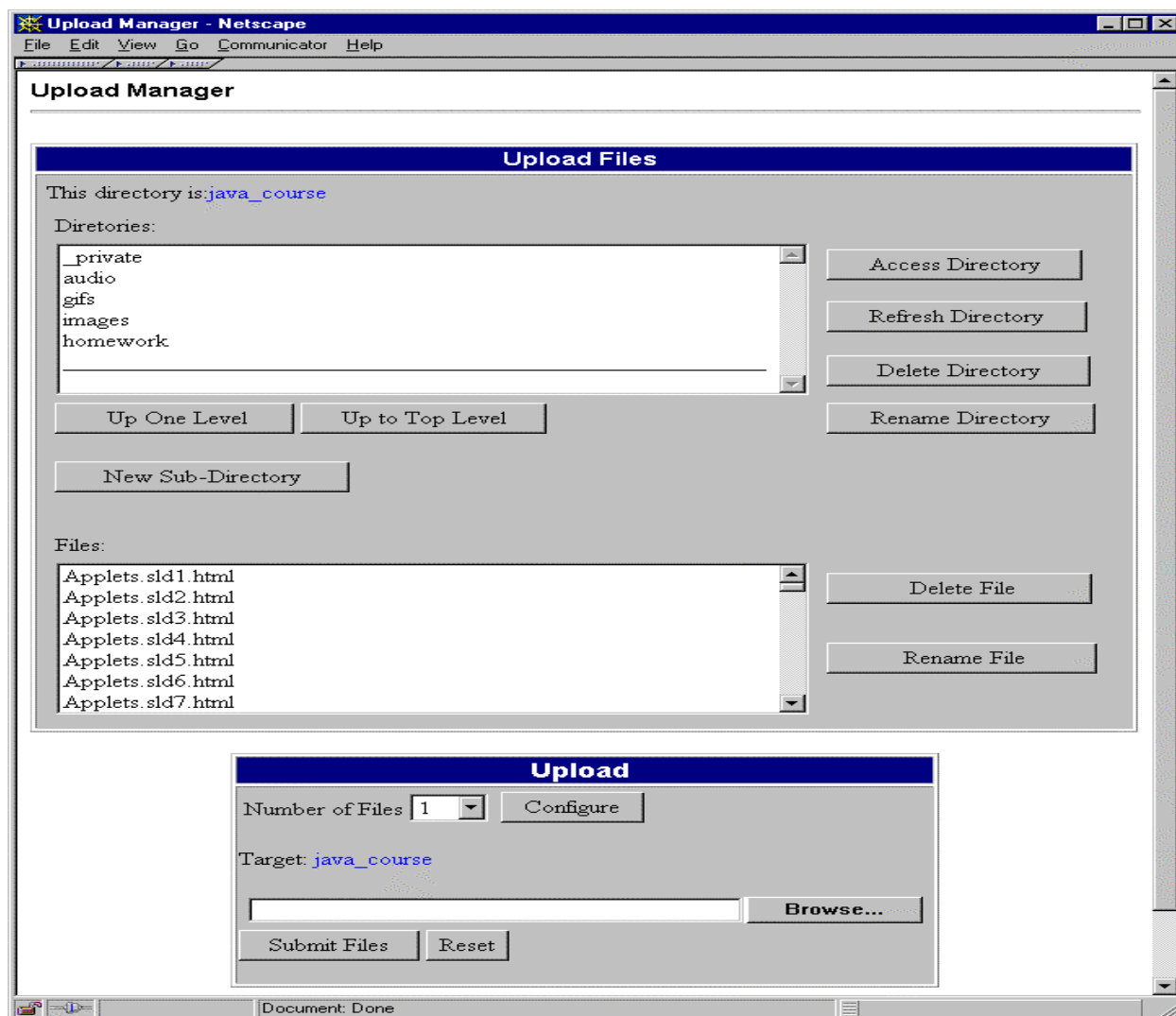
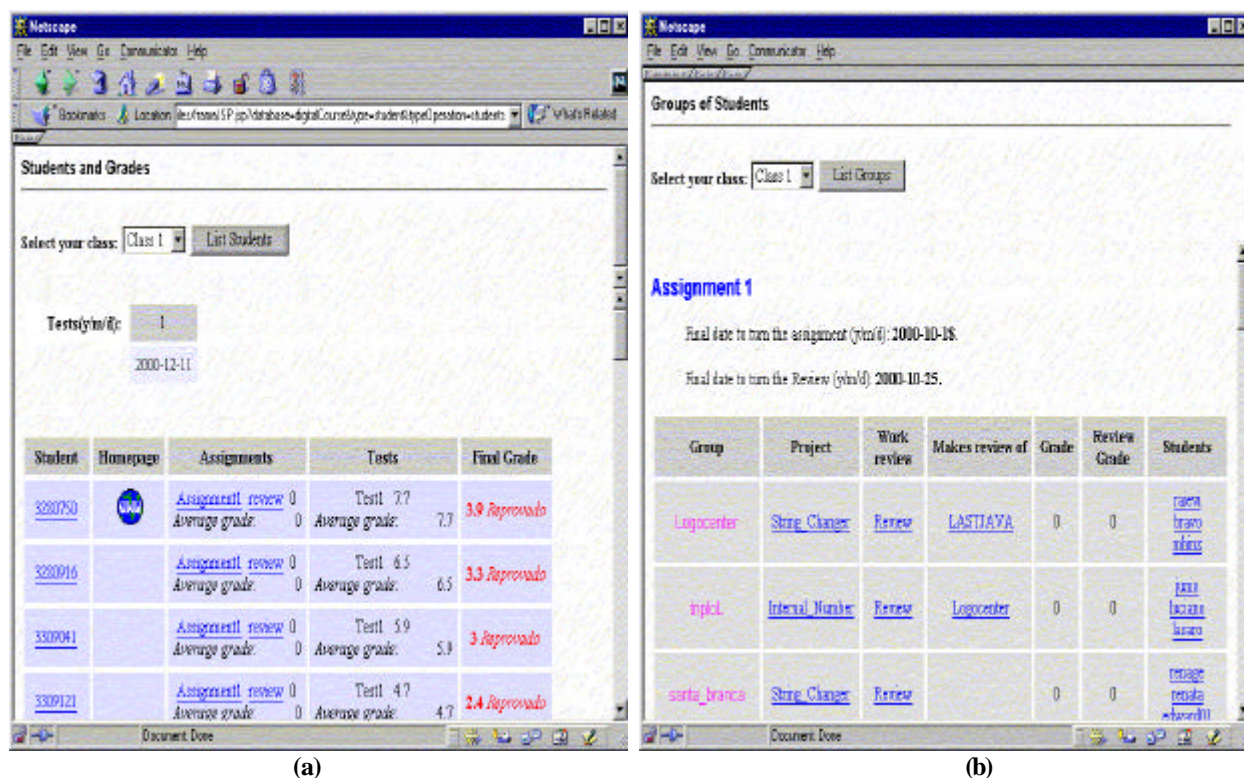


Figura 6.15 - Interface para transferência de arquivos via protocolo HTTP

Uma característica importante das ferramentas para transferência de arquivos é o controle sobre a data de entrega dos trabalhos. A ferramenta permite *upload* de arquivos de uma atividade específica até meia-noite da data estipulada no momento da criação da turma, o que permite o controle eficiente sobre as datas de entrega de trabalhos. Imediatamente após a transferência de arquivos, os trabalhos dos estudantes ou dos grupos ficam disponíveis para acesso de outros interessados.



No *applet* apresentado no *browser*, qualquer pessoa pode ter acesso às informações produzidas pelos estudantes através das opções *Students* e *Groups*. A primeira opção permite a visualização de todos os estudantes do curso separados pelas turmas, bem como a consulta às notas e aos trabalhos de cada estudante. A segunda opção leva a um documento onde são apresentados os grupos de trabalho, os *links* para os trabalhos, e os integrantes de cada grupo, também separados por turmas. As **Figuras 6.16a** e **6.16b** apresentam as interfaces destas opções.



**Figura 6.16 – (a) Interface para visualização e acesso às informações dos estudantes do curso**  
**(b) Interface para visualização e acesso às informações dos grupos de estudantes**

Observa-se na **Figura 6.16** que as datas das atividades são apresentadas no mesmo local onde os trabalhos são disponibilizados. Na interface dos estudantes são apresentadas as datas dos *tests* e dos *reports*, por serem atividades individuais. Na interface dos grupos, são apresentadas as datas de entrega dos *assignments* e *reviews*.

A entrega de uma atividade *assignment* leva à realização de uma atividade *review*, caso tenha sido definido na criação da turma. Neste caso, o administrador deve fazer a alocação dos grupos revisores, utilizando-se para isto a ferramenta para manipulação de *reviews* apresentada na **Figura 6.17**.



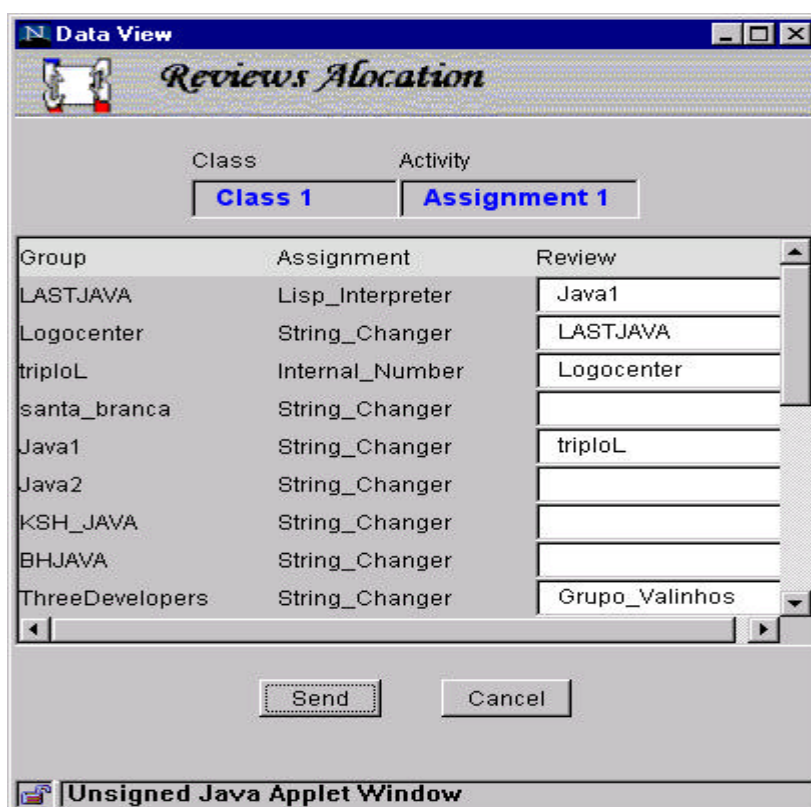


Figura 6.17 – Interface para manipulação manual de revisores de grupos de trabalho

### 6.6.5 O gerenciamento das notas dos estudantes

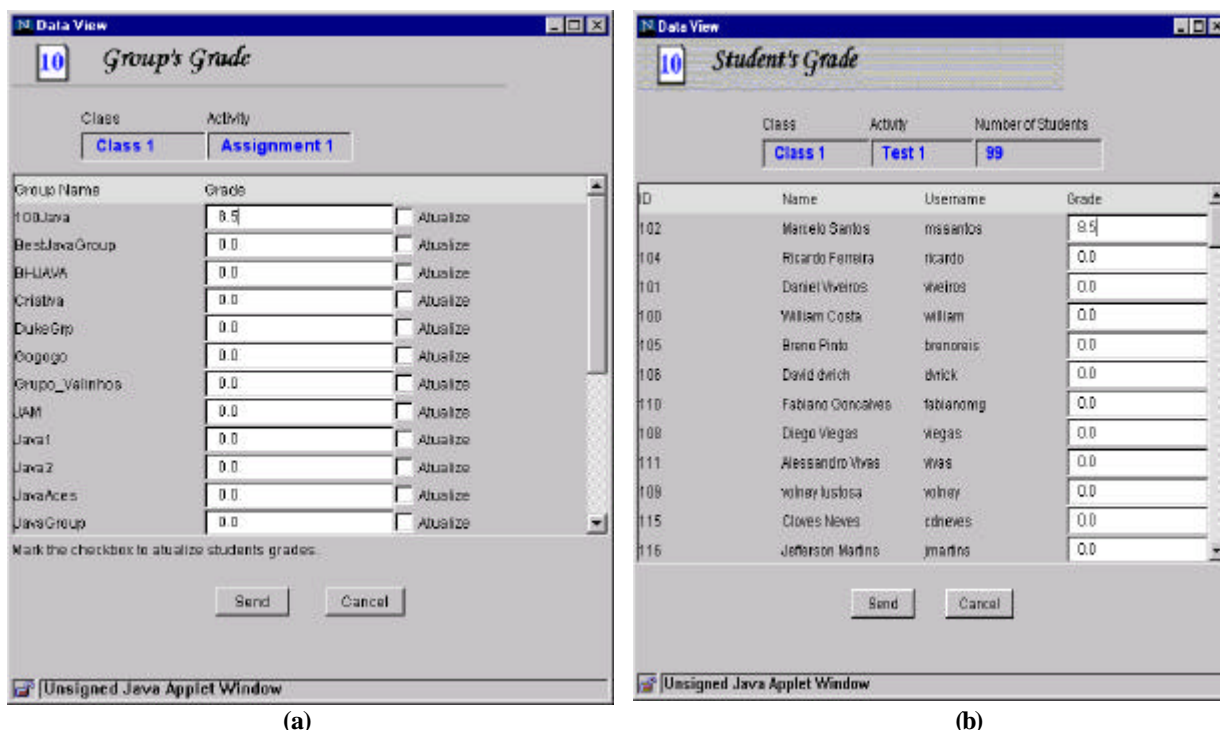
Ao completar uma atividade, os estudantes querem saber as notas obtidas por eles. Para isto, o usuário administrador ou monitor tem à sua disposição, uma ferramenta para o gerenciamento das notas.

As notas dos *assignments* e *reviews* são atribuídas aos grupos de trabalho, enquanto as notas de *reports* e *tests* são atribuídas individualmente a cada estudante. As **Figuras 6.18a** e **6.18b** apresentam as interfaces das ferramentas para manipulação das notas.

Na página de estudantes (apresentada na **Figura 6.16a**), as notas de cada atividade são apresentadas, de modo que a nota e o resultado final são automaticamente calculados.

O cálculo das médias e/ou conceitos apresentados na página do estudante é feito com base na forma de avaliação escolhida pelo administrador na criação do ambiente. Quando o administrador opta pela avaliação por conceitos, na apresentação das notas dos estudantes o WebCoM utiliza os pesos associados a cada atividade para calcular o valor total da média de

cada estudante (média ponderada). Se o administrador escolher a avaliação por média aritmética, os pesos individuais de cada atividade são ignorados, e o WebCoM calcula a média aritmética das notas, sendo que a média para ser aprovado ou reprovado é o valor definido na criação do ambiente (ver **Figura 6.7b**).



**Figura 6.18 – (a) Interface para atribuição de notas aos grupos de trabalho  
(b) Interface para atribuição de notas aos estudantes individualmente**

### 6.6.6 A comunicação entre os usuários

No WebCoM, o principal fórum de discussão entre os usuários é o *NewsGroup*. Nesta ferramenta, todos os usuários do WebCoM (incluindo pessoas interessadas em colaborar com o curso) podem enviar e receber mensagens.

Através de um formulário HTML, os usuários podem postar mensagens que, quando enviadas ao *News Agent*, são automaticamente distribuídas para todos os usuários cadastrados em um curso específico. As mensagens chegam na caixa postal do usuário com um *subject* específico do *NewsGroup*, de modo que possam ser facilmente identificadas.

Também num formulário HTML, os usuários podem ter acesso a todas as mensagens postadas e ainda, responder a alguma mensagem. A resposta é enviada a todos os usuários novamente.

## 6.7 Utilizando o WebCoM

As disciplinas de Sistemas Operacionais II e Elementos de Lógica Digital, do curso de Ciência da Computação da Universidade de São Paulo – ICMC-USP, ministrada pelo Prof. Dr. Dilvan de Abreu Moreira, estão tendo suas atividades gerenciadas pelo WebCoM desde agosto de 2000.

Além destes dois cursos presenciais, o WebCoM está sendo testado para o gerenciamento das atividades didáticas de um curso que está realizado no ICMC-USP, com estudantes espalhados por todo o Brasil, tratando-se de um treinamento à distância iniciado em setembro de 2000. Este curso – Programação Distribuída Utilizando a Linguagem Java – é parte de um projeto de mestrado que tem como um dos objetivos a avaliação do WebCoM em ambientes de educação à distância (Rosa, 2000). A **Tabela 6.1** apresenta as informações de cada curso no qual o WebCoM está sendo testado e avaliado.

Curso	Número de estudantes	Número de grupos	Número de Assignments	Número de Reports	Número de tests	Número de turmas
Sistemas Operacionais II	48	15	1	7	1	1
Elementos de Lógica Digital	47	8	1	0	1	1
Programação Distribuída Utilizando a Linguagem Java	85	28	3	0	1	2

**Tabela 6.1 – Relação entre os cursos em que o WebCoM está sendo testado**

O objetivo principal dos testes no WebCoM é eliminar os erros técnicos de implementação, e até mesmo operacionais. Inicialmente, optou-se pelos testes em um ambiente de educação presencial, como apoio ao professor no gerenciamento das atividades didáticas. A proximidade dos estudantes que estavam fazendo uso das ferramentas foi de importância essencial, uma vez que vários erros puderam ser prontamente identificados e corrigidos com a colaboração da maioria dos usuários.

Após o teste inicial com estudantes do ICMC, o WebCoM foi levado para o ambiente da Internet e passou a ser testado por usuários de todo o país. Nos três cursos gerenciados pelo WebCoM, várias informações já foram coletadas e podem ser visualizadas no endereço <http://java.icmc.sc.usp.br>. Algumas sugestões têm sido propostas pelos usuários do WebCoM, e encontram-se documentadas no **Capítulo 6**.

## **6.8 Considerações Finais**

Conforme citado, a manipulação e a divulgação das informações produzidas durante a realização de um curso são tarefas muitas vezes complexas e que envolvem a mão-de-obra direta de professores e monitores. Utilizando os recursos da Internet é possível otimizar essas tarefas através de ferramentas específicas de gerenciamento de atividades, como é o caso das ferramentas implementadas no contexto deste trabalho.

O WebCoM constitui uma forma de aproximar os estudantes de um curso oferecido a distância, utilizando os próprios serviços da Internet para prover mecanismos para coletar e disponibilizar informações.

Como nos cursos presenciais do ICMC-USP, o WebCoM pode também ser utilizado como apoio às tarefas didáticas dentro das salas de aula (Silva et al., 2000).

### 7.1 Considerações iniciais

Atualmente, a educação a distância tem sido vista como opção para atender a uma demanda cada vez mais crescente em termos do acesso a informações e ao conhecimento. Em sintonia com o desenvolvimento tecnológico, esta modalidade de ensino evolui ao longo do tempo, notadamente em termos da qualidade dos programas de educação a distância.

A Internet aparece neste meio como um importante canal para a educação a distância, uma vez que proporciona alto potencial de interação entre seus usuários. O ambiente WWW é apontado como uma poderosa ferramenta para distribuição de informação. Neste contexto, pesquisas que investigam o ambiente WWW da Internet para o suporte à educação a distância têm sido vistas com simpatia pela comunidade científica.

Neste capítulo são apresentadas as contribuições científicas que se espera obter com o desenvolvimento deste trabalho, bem como as sugestões para sua continuação.

### 7.2 Contribuições

Com a expansão da Internet e com o interesse crescente pela educação através do ambiente WWW, a integração entre as aplicações e a independência de plataforma tornam-se características cada vez mais necessárias. Em alguns ambientes de educação baseados na Internet, percebe-se que professores e estudantes ficam limitados ao conjunto de ferramentas disponíveis naquele ambiente específico, sem a possibilidade de utilizar ferramentas de outras aplicações.

Este trabalho apresentou a criação de mecanismos independentes que permitam o gerenciamento das atividades didáticas de um curso, seja à distância ou presencial, sem se prender a nenhum ambiente

específico, no que se refere à autoria e disponibilização. O WebCoM é um exemplo desta iniciativa, no qual o professor tem a possibilidade de criar e gerenciar os materiais didáticos do curso no ambiente que se sentir mais à vontade, e depois utilizar as ferramentas do WebCoM para gerenciar as atividades propostas em seu curso (Silva & Moreira, 2000c; Silva & Moreira, 2000d).

Exemplo típico desta necessidade é a criação de programas de treinamento, seja empresarial ou acadêmico, que não se adaptem aos modelos de cursos dos ambientes mais completos. Em apresentação recente na COMDEX/2000 (Moreira & Silva, 2000a) em São Paulo, o WebCoM despertou grande interesse por parte de empresários que trabalham com auditoria interna e/ou treinamento. Segundo estes empresários, utilizar um ambiente que é propício à educação acadêmica não corresponde exatamente ao tipo de necessidade que eles têm. Através das suas características, o WebCoM pode ser facilmente adaptado a uma grande variedade de situações, no campo empresarial ou acadêmico.

Apesar da proposta inicial deste projeto ser o gerenciamento de cursos a distância, com os testes realizados pôde-se comprovar a importância de mecanismos de gerenciamento mesmo em programas de educação presencial. Neste caso, as ferramentas são inseridas no ambiente de educação como ferramentas de apoio ao professor como apresentado em (Silva et al., 2000). Este é outro caso em que ambientes mais completos, tais como WebCT e AulaNet podem não se encaixar adequadamente às necessidades dos usuários.

A idéia de coletar e distribuir a informação produzida em um curso é importante para a construção do conhecimento tanto dos estudantes do curso quanto de outros interessados. Em alguns ambientes de educação via Internet, esta informação também é coletada, mas poucas vezes ficam disponíveis para o acesso por pessoas alheias ao curso. As ferramentas do WebCoM tornam possível coletar resultados das tarefas dos estudantes e disponibilizar estes resultados com uma característica adicional, que são as revisões dos próprios estudantes. Através desta característica, é possível se ter uma avaliação, mesmo que simplificada, do material a que se tem acesso. Além disso, esta característica oferece auxílio ao professor na avaliação dos trabalhos dos estudantes. No

gerenciamento de cursos a distância, esta característica é essencialmente importante, uma vez que o número de estudantes pode ser grande e conseqüentemente, consumir um tempo maior do professor.

Por fim, o conceito de *views* de dados (*mobile views*), apresentado neste trabalho, contribui diretamente para o desenvolvimento mais eficiente e organizado de aplicações que envolvem a comunicação entre cliente/servidor no ambiente da Internet (Moreira & Silva, 2000b). A característica principal das *views* de dados, que é a implementação dos código de acesso à base de dados e da interface gráfica no mesmo local, possibilita a facilidade de manutenção nos códigos de uma grande aplicação. Além disso, a criação de um canal de comunicação seguro torna o uso das *views* de dados essencialmente importante para a transmissão de dados através da Internet.

### 7.3 Sugestões para Trabalhos Futuros

Como continuidade para o desenvolvimento do WebCoM, uma das principais tarefas é a conversão da base de dados relacional, atualmente em SQL, para uma base de dados em documentos XML. A hierarquia dos documentos XML oferece recursos para o armazenamento e recuperação de informações, o que expande as possibilidades de intercâmbio de documentos em aplicações hipermídia distribuídas.

Nos testes realizados, principalmente no curso a distância onde o WebCoM está sendo testado (apresentado no **Capítulo 6**), notou-se também uma necessidade de desenvolvimento ou integração de ferramentas de comunicação, tais como *chats*, *White boards*, dentre outras.

As interfaces do WebCoM podem ser melhoradas e versões em Português e em outros idiomas podem ser geradas através das características de internacionalização da linguagem Java.

Uma necessidade que foi claramente identificada durante os testes do WebCoM é o desenvolvimento de mecanismos que permitam a consulta a todas as informações armazenadas nas bases de dados e nos diretórios dos estudantes. O WebCoM, apesar de ter alguns mecanismos para consulta, tem limitações na apresentação da informação organizada (por exemplo, possibilidade de listagens por nomes, por *usernames*, por número de identificação, dentre outras informações).

Um aspecto que também merece destaque e que pode ser estudado é a criação de um mecanismo para gerar e integrar formulários específicos para cada tipo de curso que se deseja gerenciar. O formulário para cadastro de candidatos tem limitações na aquisição de informações específicas, que podem ser importantes na avaliação da ficha de cadastro do candidato.

#### **7.4 Considerações Finais**

Este capítulo apresentou as conclusões deste trabalho, bem como suas contribuições para a comunidade e as sugestões para a sua continuação.

A principal contribuição deste trabalho foi o desenvolvimento do conjunto de ferramentas de gerenciamento, que formam o WebCoM. De modo geral, pode-se dizer que com o uso do WebCoM o professor pode se tornar menos sobrecarregado na realização de um curso via Internet, economizando tempo em atividades como o controle sobre os grupos de trabalho, a distribuição das notas aos alunos (a ferramenta realiza os cálculos tanto de médias quanto de conceitos aplicados às médias, inclusive com pesos diferenciados para cada atividade), a divulgação dos resultados do curso através dos trabalhos entregues pelos estudantes, dentre outras.

Espera-se com este trabalho que novas ferramentas sejam produzidas seguindo-se a premissa do WebCoM que é a independência, tanto de plataforma quanto de aplicações, e que os resultados possam significar contribuições efetivas para incrementar o uso da Internet em aplicações educacionais.



## REFERÊNCIAS BIBLIOGRÁFICAS

- (Abowd et al., 1999) ABOWD, G.; PIMENTEL, M.; KERIMBAEV, B.; ISHIGURO, Y. e GUZDIAL, M.: Anchoring discussions in lecture: an approach to collaboratively extending classroom digital media, *Proceedings of CSCL'99*, Palo Alto, CA, december. 1999.
- (Abowd, 1999) ABOWD, G.: Classroom 2000: an experiment with the instrumentation of a living educational environment. *IBM Systems Journal*, v. 38, n.4, p.508-530, october. 1999.
- (AulaNet, 1997) PROJETO AULANET. WWW: Disponível *on-line*, <http://aulanet.les.inf.puc-rio.br/aulanet/>. Visitado em março de 1999.
- (AulaNet, 1998) PROJETO AULANET. Ajudando Professores a Fazer seu Dever de Casa. WWW: Disponível *on-line*, <http://aulanet.les.inf.puc-rio.br/aulanet>. Visitado Junho de 1999.
- (Barker, 1992) BARKER, P. : *Computer-Based Training: An Institutional Approach*. Education & Computing, 1992.
- (Bellovins, 1989) BELLOVINS, S. M.: Security Problems in the TCP/IP Protocol Suite. *Computer Communications Review*, v.9,n.2,pp.32-48, abril. 1989. WWW: Disponível *on-line* em: <http://www.cert.lu/security/documents.html>. Visitado em setembro de 2000.
- (Bray et al, 1997) BRAY, T. et al.: Extensible Markup language (XML) – XML Principles, Tools and Techniques. *World Wide Web Journal*, v.2, n.4, p.29-66, 1997.
- (Brown, 1989) BROWN, H.: Standarts for Structured Documents. *The Computer Journal*, v.32, n.6, p.505-514, december, 1989.
- (Bufford, 1994) BUFORD, J.: *Uses of Multimedia Information in Multimedia Systems*, Addison-Wesley, 1994.
- (Burdea & Coffet, 1993) BURDEA, G.: and COIFFET, P. *Virtual Reality Technology*. John Wiley & Sons Inc, 1993.
- (Chess, 1999) CHESS, D. ; HARRISON, C. e KERSHENBAUM, A.: Mobile Agentes: Are They a Good Idea?, *IBM Research Report*, WWW: disponível *on-line* em <http://www.research.ibm.com/iagentes/paps/mobile-idea.ps>. Visitado em junho de 1999.

- (Colla, 1999) COLLA, E.C. : Servlet, java do lado do servidor. WWW: Disponível on-line em <http://www.insite.com.br/docs/develop-servlet95.html>. Visitado em abril de 1999.
- (Comer, 1995) COMER, D. E.: *Internetworking whit TCP/IP*. 3ed., Prentice-Hall, 1995.
- (Connoly, 1997) CONNOLOY, D.: XML Principles, Tools and Techniques. *World Wide Web Journal*. O'Reilly. v.2, Issue 4, 1997.
- (Cornell & Horstman, 1997) CORNELL, G. e HORSTMANN, C.S.: *CORE JAVA – Guia Autorizado da SUN Microsystems*, São Paulo, Makron Books, 1997.
- (Coulloris et al., 1994) COULOURIS, G.; DOLLIMORE, J. and KINDBERG, T.: *Distributed Systems: Concepts and Desing*. 2ed., Addison-Wesley Publishing Company, 1994.
- (Cryptix, 1999) Cryptix Homepage. WWW: Disponível on-line em <http://ai.cryptix.org/index.html>. Visitado em fevereiro de 1999.
- (Cyclades, 1996) CYCLADES Corporation: *Guia Internet de Conectividade*. Cyclades Brasil, 1996.
- (Damasceno Jr., 1996) DAMASCENO JR., A.: *JAVA: Programação para a Internet*. 3.ed. São Paulo, Érika, 1996.
- (Franklin & Graesser, 1996) FRANKLIN, S. e GRAESSER, A.: *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents*. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996. WWW: Disponível on-line, [http://agents.umbc.edu/Publications\\_and\\_presentations/Recommended\\_Papers/](http://agents.umbc.edu/Publications_and_presentations/Recommended_Papers/). Visitado em junho de 1999
- (Freire & Nunes, 1999) FREIRE, M.E.P. e Nunes, M.G.V.: O Sistema Tutor de um Ambiente Inteligente para Treinamento e Ensino. *Anais do IV Simpósio de Teses e Dissertações Concluídas - ICMC-USP*, p.260-269, São Carlos, 1999.
- (Fucks & Lucena, 2000) FUCKS, H. e Lucena, J.C.P. de: Aprendizagem e trabalho Cooperativo no Ambiente Aulanet. *Monografias em Ciência da Computação*, n.11/00. WWW: Disponível on-line, <http://aulanet.les.inf.puc-rio.br/aulanet/>. Visitado em agosto de 2000.
- (Guzdial, 1999) GUZDIAL, M.: Collaborative website supporting open authoring, Enviado ao *Journal of the Learning Sciences*, 1999
- (Hall, 2000) HALL, M.: *Core Servlets and Java Server Pages, 1ed.*, Prentice Hall, 2000.
- (Harrison, 1996) HARRISSON, M.: *The Essentials Elements of Hypermedia*. Academic Press, 1996.

- (Holmberg, 1977) HOLMBERG, B.: *Distance education: A survey and bibliography*, London, 1977.
- (HTML, 1992) HTML Specification. WWW: Disponível *on-line*, <http://www.w3.org/MarkUp>. Visitado em novembro de 1998.
- (HTML, 1997) HTML Tutorial. WWW: Disponível *on-line*, <http://www.icmc.sc.usp.br/manuals/HTML/intro.html>. Visitado em novembro de 1998.
- (HTTP, 1992) HTTP Specification. WWW: Disponível *on-line*, <http://www.w3.org/Protocols/HTTP/HTTP2.html>. Visitado em novembro de 1998.
- (ISO, 1986) ISO/IEC IS 8879. *Information Processing - Text and Office Systems – Standards Generalized Markup Language (SGML)*, 1986.
- (ISO, 1997) ISO/IEC JTC1/SC29/WG11 N1909, Overview of the MPEG-4 Version 1 Standard, 1997. WWW: Disponível *on-line* em <http://drogo.cselt.stet.it/mpeg/public/w1909.htm>. Visitado em setembro de 2000.
- (Jamsa et al., 1997) JAMSA, K.: et al. *JAVA: A Biblioteca do Programador*. São Paulo, Makron Books, 1997.
- (JavaWorld, 2000) JAVAWORLD Homepage. Using XML and JSP together. WWW: Disponível *on-line*, <http://www.javaworld.com/javaworld/jw-03-2000/jw-0331-ssj-jspxml.html>. Visitado em agosto, 2000.
- (Jennings & Wooldrige, 1994) JENNINGS, N. R. e WOOLDRIGE, M. : Agent Theories, Architectures, and Languages: a Survey. *Proceedings ECAI – Workshop on Agent Theories, Architecture and Languages*, Amsterdam, p.1-32, 1994.
- (Jepson, 1997) JEPSON, B.: *Dominando JAVA*. São Paulo, Makron Books, 1997.
- (Johnson, 1999) JONHSON, M.: (1999, Abril). XML for the Absolute Beginners. WWW: Disponível *on-line*, <http://www.javaworld.com/javaworld/>. Visitado em março de 1999.
- (Keegan, 1991) KEEGAN, D.: *The Foundations of the Distance Education*. 2.ed. London, Routledge, 1991.
- (Kerberos, 1999) KERBEROS's Tickets. WWW: Disponível *on-line* em <http://www.cit.cornell.edu/computer/connect/security/kerberos.html>. Visitado em dezembro de 1999.
- (Ketchpel & Genesereth, 1994) KETCHPEL, S. e GENESERETH, M.: Software Agents. *Communications of ACM*, v.37, n.7, p.48-53, julho, 1994.
- (Lemair & Shae, 1997) LEMAIR, M.H.W e SHAE, Z.Y.: VideoConferencing over Packet-Based Networks. *IEEE JSAC*, v.15, n.1, august, 1997.

- (Lucena et al., 2000) LUCENA, J.C.P. de et al.: Tecnologia de Informação Aplicada à Educação: Um (Meta) Curso no Ambiente AulaNet. *Monografias em Ciência da Computação*, n.17/00. WWW: Disponível *on-line*, <http://aulanet.les.inf.puc-rio.br/aulanet/>. Visitado em agosto de 2000.
- (Lucena, 1997) LUCENA, J.C..P. de: Curso sobre Sociedade da Informação. WWW: Disponível *on-line*, <http://www.les.inf.puc-rio.br/socinfo>. Visitado em novembro de 1998.
- (Macedo et al., 1999) MACEDO, A. A. et al.: StudyConf: infra-estrutura de suporte ao aprendizado cooperativo na WWW. *Revista Brasileira de Informática na Educação*, n.5, p. 77-102, 1999.
- (Macedo, 1999) MACEDO, A. A.: *Utilização de Tecnologias Hipermídia e de Cooperação para apoiar o Ensino*. São Carlos, 1999. 89p. Dissertação (Mestrado) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- (Maes, 1994) MAES, P.: Agents that Reduce Work and Information Overload. *Communications of the ACM*, v.37, n.7, p.30-40, julho, 1994.
- (Microsoft, 1999) MICROSOFT Homepage. WWW: Disponível *on-line*, <http://www.microsoft.com>. Visitado em novembro de 1998.
- (Moore & Kearsley, 1996) MOORE, M.G. e KEARSLEY, G.: *Distance Education: A System View*, Wadsworth Publishing, 1996.
- (Moore et al., 1990) MOORE, M.G. ; THOMPSON, M.M.; QUIGLEY, A.B.; CLARK, G.C.; e GOFF, G.G. : *The effects of distance learning: A summary of the literature*, 1990.
- (Moreira & Walczowski, 1997) MOREIRA, D. A. e WALCZOWSKI, L. T.: Using Software Agents to Generate VLSI Layouts. *IEEE Expert Systems - Intelligent Agents*, v.12,n.6, p.26-32. Novembro/dezembro de 1997.
- (Moreira et al., 1995) MOREIRA, E. S.; NUNES, M.G.V. e PIMENTEL, M.G.C.: Design Issues for a Distributed Hypermedia-Based Tutoring Systems (HyDTS). *Proceedings of the International Conference on Computer Application in Industry*, p.108-113, dezembro de 1995.
- (Morrison et al., 1998) MORRISON, M. et al.: *JAVA Unleashed*. 3.ed. Sams.net Publishing, 1998.
- (Morrison et al., 1999) MORRISON, M. et al.: *Como Programar em JAVA Beans*. São Paulo, Makron Books, 1999.
- (Netscape, 1998) NETSCAPE Homepage. WWW: Disponível *on-line*, <http://netscape.com>. Visitado em novembro de 1998.
- (Neuman & Ts'o, 1994) NEUMAN, B. C. e TS'O Theodore: Kerberos: An Authentication Service for Computer Networks, *IEEE Communications*, v.32, n.9, p.33-38, september, 1994.

- (Nunes et al., 1997) NUNES, M.G.V. et al.: SASHE: Autoria de Aplicações Hiperídia para o Ensino. *Anais do VIII Simpósio Brasileiro de Informática na Educação*, v.1, p.425-440, São José dos Campos, novembro, 1997.
- (Nunes, 1994) NUNES, I. B.: Noções de Educação a Distância. *Revista Educação a Distância*, n. 4/5, Dez./93-Abr/94, pp. 7-25, Brasília, Instituto Nacional de Educação a Distância, 1994.
- (Pimentel et al., 1998) PIMENTEL, M.G.C.; KUTOVA, M.A.S.; MACEDO, A.A.; FORTES, R.P.M. e TEIXEIRA, C.A.C.: Hiperdocumentos Estruturados no Suporte ao Trabalho Cooperativo em Sistemas Abertos Distribuídos. *Anais do XXV Seminário Integrado de Software e Hardware*, Belo Horizonte, p.158-173, agosto, 1998.
- (Pimentel et al., 2000) PIMENTEL, M.G.C.; ABOWD, G.; ISHIGURO, Y.; KERIMBAEV, B. e GUZDIAL M. : Supporting long-term educational activities through dynamic Web interfaces, In. *Interacting with Computers Journal*, 2000.
- (Pires & Pimentel, 2000) PIRES, D. F. e PIMENTEL, M.G.: Definição, Geração e Apresentação de Documentos Estruturados Didáticos na WWW, *Anais do V Simpósio de Teses e Dissertações em Andamento da ICMC-USP*, pp.67-68, São Carlos, 2000.
- (Robie et. al., 1998) ROBIE, J. et al. : (1998, Setembro). XML Query Language (XQL). WWW: Disponível on-line, <http://www.w3.org>. Visitado em julho de 1999.
- (Romiszowski, 1993) ROMISZOWSKI, A.: Telecommunications and Distance Education. *ERIC Digest. ERIC Clearinghouse on Information Resources Syracuse NY*. WWW: Disponível on-line, [http://www.ed.gov/databases/ERIC\\_Digests/ed358841.html](http://www.ed.gov/databases/ERIC_Digests/ed358841.html). Visitado em setembro, 2000.
- (Romiszowski, 2000) ROMISZOWSKI, A.: Educação a Distância : Passado, Presente e Futuro (*slides do PowerPoint*) Alfenas – MG, UNIFENAS/ABED – *Seminário Integrado de Educação a Distância*, setembro, 2000. 36 slides.
- (Rosa, 2000) ROSA, V.F.da: Uma metodologia para um curso a distância na Internet que promova o trabalho cooperativo, *Anais do V Workshop de Teses e Dissertações em Andamento do ICMC-USP*, São Carlos, 2000.
- (Santos Jr, 1998) SANTOS JR, J.B. dos.: *Documentos Estruturados para o Domínio de Aplicação Ensino: Modelagem, Autoria e Apresentação no Ambiente WWW*. São Carlos, 1998. 107p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- (Scapin & Garcia Neto, 1997) SCAPIN, R. H. e GARCIA NETO, A.: Desenvolvimento de uma Ferramenta para Criação e Correção Automática de Provas na World Wide Web. *Anais do VIII Simpósio Brasileiro de Informática na Educação*, v.1, p.593-608, São José dos Campos, novembro, 1997.

- (Silva & Moreira, 2000a) MOREIRA, D. A. & SILVA, E. Q. da: Ferramentas para Gerenciamento de Cursos via Internet. *Pôster apresentado na Comdex Sucesu'2000 - 21 a 25 de agosto*, São Paulo, 2000. WWW: Disponível on-line em <http://agentsresearch.com>.
- (Silva & Moreira, 2000b) MOREIRA, D. A. & SILVA, E.Q.: Java Mobile Data Views, *artigo submetido (avaliação) para a Internet Computing*, 2000.
- (Silva & Moreira, 2000c) SILVA, E.Q.da e MOREIRA, D.A: Uso de Agentes de Software para Gerenciamento de Cursos a Distância via Internet, *Anais do Workshop de Informática na Educação*, Curitiba, julho, 2000.
- (Silva & Moreira, 2000d) SILVA, E.Q.da e MOREIRA, D.A: Use of Software Agents to the management of Distance Education Courses over the Internet, *Proceedings of the ICECE*, São Paulo, august, 2000.
- (Silva et al., 2000) SILVA, E.Q.da; MOREIRA, D.A.; SANTOS JR, J.B.dos: Computadores no Ensino: Uma abordagem voltada para o Suporte aos Professores no Desenvolvimento de Atividades Didáticas. *Anais do Simpósio Brasileiro de Informática na Educação*, Maceió, novembro, 2000.
- (SMIL, 1999) SMIL Specification. WWW: Disponível on-line em: <http://www.circuitfunk.com/haveasmil/smil.htm>. Visitado em junho de 2000.
- (Sun, 1999) Sun's Homepage. WWW: Disponível on-line, <http://www.sun.com>. Visitado em novembro de 1999.
- (Tanenbaum, 1996) TANENBAUM, A.S.: *Distributed Operating Systems*, Prentice Hall, 1996)
- (Taylor, 1997) TAYLOR, A.: *JDBC Developer's Resource*, Prentice Hall, 1997.
- (Thomas et al., 1997) THOMAS, M.D. et al.: *Programando em JAVA para a Internet*. São Paulo, Makron Books, 1997.
- (TopClass, 1999) TOPCLASS's HomePage. WWW: Disponível on-line, <http://www.wbtsystems.com/spotlight/gallery.html>. Visitado em março de 1999.
- (Valente, 2000) VALENTE, J. A. Diferentes usos do Computador na Educação. WWW: Disponível on-line em <http://www.proinfo.br>. Visitado em agosto de 2000.
- (WebCT, 1999) WEBCT Homepage. WWW: Disponível on-line, <http://www.webct.com/webct>. Visitado em março de 1999.