# SynchroTalk: Uma Ferramenta Síncrona para Ensino a Distância

Berto de Tácio P. Gomes[1]
Omar Andres C. Cortes[2]

Centro Federal de Educação Tecnológica do Maranhão (CEFET-MA)
Departamento Acadêmico de Informática (DAI)
Av. Getúlio Vargas, 04 - CEP 65000-000 São Luis-MA
[1]bertodetacio@gmail.com
[2]omar@cefet-ma.br

**Resumo.** O objetivo deste trabalho é apresentar uma ferramenta síncrona para auxílio no ensino a distância. Nesse contexto, a sincronização é utilizada para coordenar a comunicação entre membros de uma reunião virtual (chat). Os algoritmos de sincronização implementados são baseados na sincronização utilizada em sistemas distribuídos. Três algoritmos foram implementados: centralizado, token-ring e distribuído. Serão discutidos os aspectos tecnológicos de implementação e os aspectos pedagógicos envolvidos.

**Palavras-Chave:** Sistemas Distribuídos, Sincronização, Ensino à Distância.

# SynchroTalk: A Synchronous Tool for Distance Education

**Abstract.** The purpose of this work is to show a synchronous tool for helping the distance education. The synchronization is used to coordinate virtual meetings, in this context. The implemented algorithms are based on distributed systems synchronization. Three algorithms have been implemented: centralized, token-ring and distributed. Technical and Pedagogical aspects have been discussed, as well.

**Keywords:** Distributed Systems, Synchronization, Distance Education

(Received Julho, 2006 / Accepted Fevereiro, 2007)

## 1 Introdução

Atualmente, o mercado de trabalho esta bastante exigente. A cada dia que passa as empresas requerem profissionais cada vez mais competentes e treinados. Entretanto, a maioria dos cursos de treinamento encontra-se nas grandes capitais, pois de maneira geral é dispendioso manter cursos e professores em cidades mais afastadas cuja população é relativamente pequena.

O ensino a distância ou *Educação a Distância* surgiu com o objetivo de atender essas comunidades sem a necessidade de deslocar professores para as cidades mais afastadas. Na verdade, a educação a distância começou a ser bem vista na década de 50, quando o principal meio de comunicação era o correio tradicional [4]. Com o advento dos computadores e das redes de comunicação, as possibilidades para efetivar e incentivar o ensino a distância aumentaram significativamente. De maneira geral, os gastos passam a ser apenas com equipamentos na cidade origem da instituição e com o desenvolvimento do material didático.

Dependendo da forma como se dá o ensino à distância, muitas vezes os gastos com equipamentos nas cidades afastadas é zero, pois o curso pode ser acompanhado pelo estudante a partir do seu próprio computador, na sua casa ou escritório. Outro

ponto positivo a destacar é que o estudante pode acompanhar o curso nos horários que mais lhe convier.

Apesar da flexibilidade de horários e de local provenientes dos cursos à distância, encontros periódicos através da Internet podem também ser solicitados. Nesse caso, uma das ferramentas de baixo custo mais utilizadas é o *chat*, que pode proporcionar reuniões virtuais entre alunos e professores. Entretanto, essa ferramenta pode causar sérias dificuldades quando, por exemplo, alunos passam a discutir outros assuntos em paralelo ou quando todos `"falam"` ao mesmo tempo. Quando isso ocorre, o professor tentará controlar a situação, embora isso dependa exclusivamente da colaboração dos alunos. Nesse contexto, observa-se que apesar da evolução do uso da tecnologia na educação, a sua aplicação não é garantia de qualidade [12].

Sendo assim, observa-se que este tipo de ferramenta carece de mecanismos que coordenem o acesso ao recurso de envio e recebimento de mensagens. Oeiras [8], Pimentel [9], entre outros, mostram as dificuldades encontradas em ferramentas de bate-papo quando vários usuários enviam mensagens simultaneamente. Dessa forma, percebe-se nitidamente a necessidade de controlar o acesso ao recurso de envio de mensagens.

No contexto de sincronização, os trabalho de Smith [10] e Pimentel [9] utilizam *threads* para sincronizar as conversas, onde o usuário deve clicar na mensagem que deseja responder. Essa metodologia demonstrou-se ineficiente, pois os usuários tiveram dificuldades em se adaptar à nova forma de chat. Em Vahl Junior [15], apresentam-se modelos para organizar as conversas, sendo esses modelos: assembléia, seminário, tradicional e personalisado. Os modelos assembléia e seminário exigem que o usuário peça a palavra e sua requisição é colocada em uma fila. A forma tradicional é a mesma dos chats comuns, enquanto que a personalisada mistura os dois primeiros modelos. O problema apresentado por este trabalho é que os usuários ficavam perdidos com relação ao seu próprio posicionamento na fila. No trabalho de OEIRAS [7] é introduzida a figura do coordenador para sincronizar a comunicação. Na maioria dos casos pesquisados, observa-se que os participantes têm dificuldades na utilização do chat, pois os mecanismos utilizados não são transparentes.

Observa-se então, que o problema de sincronização em chats é semelhante ao encontrado em sistemas distribuídos (SD) [3] para o tratamento de exclusão mútua [14], ou seja, vários processos tentando acessar o mesmo recurso (região crítica). No caso do ensino a distância, a região crítica é considerada como sendo o ato de `"falar"` na ferramenta de chat.

Nesse âmbito, este artigo pretende contribuir com uma proposta para aplicação dos algoritmos de sincronização (encontrados na teoria dos sistemas distribuídos) em uma ferramenta de chat (servidor e cliente), denominado SynchroTalk, direcionada à educação a distância. Além disso, os mecanismos implementados são transparentes aos usuários, sendo que o máximo esforço exigido do usuário é pedir a palavra.

Para cumprir o objetivo aqui estabelecido este artigo esta organizado da seguinte maneira: a Seção 2 apresenta a relação entre os algoritmos de sincronização e a ferramenta SynchroTalk; a Seção 3 aborda aspectos de implementação do chat sincronizado; na Seção 4 discute-se o comportamento dos algoritmos no chat sincronizado e seu impacto técnico e pedagógico; na Seção 5 são apresentadas as conclusões deste trabalho; finalmente, na Seção 6 são mostrados os trabalhos futuros.

## 2 Algoritmos em SD e o SynchroTalk

Os três principais algoritmos encontrados em sistemas distribuídos são: centralizado, distribuído e token-ring [13]. Neste trabalho os três algoritmos foram implementados e são apresentados nas próximas seções.

### 2.1 Centralizado

No algoritmo centralizado, um processo envia uma mensagem de solicitação de recurso ao coordenador antes entrar na região crítica. O coordenador recebe a mensagem e a coloca em uma fila de mensagens recebidas. O processo solicitante deverá aguardar a resposta do coordenador que verifica se algum outro processo ocupa o recurso solicitado. Quando um processo deixa a região critica ele envia uma mensagem ao coordenador informando que liberou o recurso, em seguida, o coordenador atenderá o próximo pedido.

No *SynchroTalk*, os participantes agem como se fossem processos. Para qualquer que seja o coordenador, pode-se determinar o tempo que os participantes terão para postarem suas mensagens. Neste caso, ao terminar o tempo, o recurso será liberado automaticamente, permitindo que o coordenador atenda outra solicitação.

### 2.2 Token-Ring

No segundo algoritmo implementado (*token-ring*), os participantes são ordenados em um anel lógico pelo qual circula um token. Quando alguém deseja falar deve reter o token e soltá-lo após enviar as mensagens, sendo que somente um token deve circular através do anel de participantes.

Para não afetar o desempenho da reunião um token deve ser retido por uma quantidade máxima de mensagens ou por um período determinado de tempo. Neste trabalho considera-se que a quantidade máxima de mensagens é igual a 1 (um).

### 2.3 Distribuído

Neste algoritmo quando um processo deseja entrar na região critica, ele envia uma mensagem com dois parâmetros (seu pedido e o tempo atual) para todos os outros participantes, incluindo para si próprio. Quando um processo recebe uma mensagem desse tipo ele reagirá de acordo com o seu estado podendo:

- Enviar uma mensagem de OK ao processo emissor caso não esteja na região crítica e não queira entrar nela;
- Não responder a mensagem recebida e inseri-la numa fila caso esteja na região crítica;
- Caso o processo deseje entrar na região crítica e ainda não o tiver feito, comparar o tempo da mensagem recebida com o tempo constante da mensagem que ele enviou aos demais processos, enviando uma mensagem de OK como resposta à mensagem recebida se esta tiver um tempo menor que a mensagem enviada; caso contrário, ele não responde a mensagem e insere-a numa fila.

Quando um participante envia seu pedido aos demais, ele só poderá ter acesso ao recurso quando receber de todos os outros uma mensagem de OK. Ao finalizar, o participante com a palavra deve enviar uma mensagem de liberação quando não quiser mais enviar mensagens (mensagens normais) ou caso o tempo de envio termine (essa possibilidade foi adotada caso se queira evitar que um participante monopolize o recurso).

## 3 Implementação

O chat foi desenvolvido em Linguagem Java, pois o emprego desta linguagem possibilita o uso de várias tecnologias auxiliares na implementação. Por exemplo, em Java permite-se a utilização de *Frameworks* que auxiliam no desenvolvimento de sistemas. No SynchroTalk utilizou-se o framework ProActive [6].

ProActive é um framework para computação concorrente, paralela, distribuída e em grade (*GRID middleware*). Entre os motivos de sua escolha pode-se citar o fato de que, ao contrário de outras tecnologias como CORBA [2] e JavaRMI [5], o ProActive *middleware* não requer que classes que originam objetos remotos sejam modificadas, simplificando o desenvolvimento de aplicações distribuídas e permitindo o polimorfismo entre objetos locais e remotos.

Vale destacar que no ProActive a comunicação distribuída ocorre através de objetos ativos. Objetos ativos são unidades de atividade que podem ser acessadas remota e concorrentemente e têm a capacidade de migrar de um computador para outro através da rede, possibilitando o uso de agentes móveis. Cada objeto ativo possui sua própria *thread*, que executa métodos invocados dentro do objeto ativo. Através da utilização do ProActive, o programador não precisa manipular explicitamente objetos de linha de execução e ainda pode definir políticas FIFO ou LIFO para a execução de chamadas. Entre outros recursos interessantes no ProActive estão os grupos de comunicação, que proporcionam melhorias de desempenho, eficiência e flexibilidade em aplicações distribuídas [1]. O middleware utilizado contém ainda mecanismos de tolerância a falhas, *checkpoint*, segurança e monitoramento. Além das vantagens mencionadas, o ProActive é open source sob licença LGPL, o que o tornou uma boa escolha em relação a tecnologias proprietárias de mesma finalidade, como por exemplo, o *middleware* Voyager [11].

Um aspecto interessante do ProActive é a ferramenta IC2D, que permite o monitoramento da aplicação em tempo de execução. Esse recurso pode ser usado em todos os mecanismos descritos. Além disso, é possível visualizar chamadas remotas, perceber falhas no sistema e ainda acompanhar o movimento do agente utilizado no mecanismo *token ring* e perceber se o mesmo está funcionando de forma eficiente. A Figura 1 mostra um servidor comunicando-se com três clientes.

Com relação à aplicação, a Figura 2 apresenta uma tela do chat de um cliente já conectado. Na figura observa-se o botão *Pedir Palavra* que o usuário deve pressionar quando quiser `falar`. Quando o mesmo obtiver a permissão o usuário poderá enviar

mensagens.

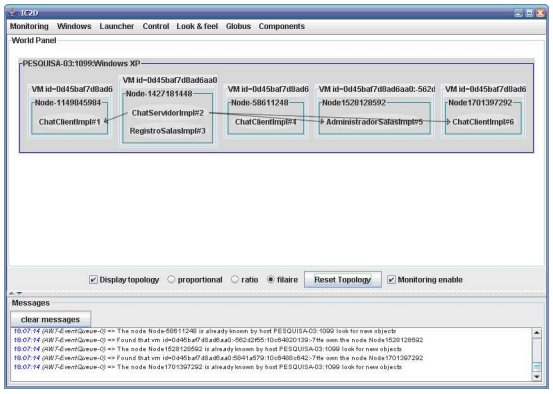


**Figura 1:** Um servidor comunicando-se com clientes.

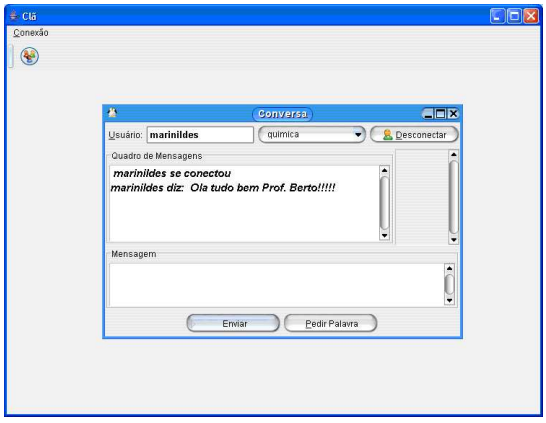


**Figura 2:** Um cliente conectado.

### 3.1 Algoritmo Centralizado

Na implementação do mecanismo centralizado, um objeto ativo (localizado no servidor) faz o papel de coordenador da região crítica. Esse objeto recebe todas as solicitações dos participantes do *Synchro-Talk* e as armazena em uma fila de mensagens. Este objeto ativo verifica se algum participante está com permissão para enviar mensagens, e atende a próxima solicitação da fila assim que o participante que estiver com a palavra, liberar o recurso.

O ProActive adiciona um conjunto de exceções que vão além das exceções normais (originadas das regras de negócio), esse novo conjunto é chamado de *Exceções Não Funcionais* (NFE), que indicam problemas durante a comunicação na rede entre objetos distribuídos. Desta forma, quando um objeto ativo percebe um problema na comunicação com o participante que estava enviando mensagens, ele pode imediatamente atender a solicitação de outro participante. Quando um tempo limite de postagem de mensagens for determinado, o coordenador atende outra solicitação assim que o tempo terminar. Um coordenador humano poderá manipular o objeto ativo de forma a fazer concessões de recurso de acordo com a conveniência, permitindo maior flexibilidade neste mecanismo.

### 3.2 Algoritmo Distribuído

No mecanismo distribuído utilizou-se o recurso de formação de grupos de comunicação, que permitiu reunir todas as referencias remotas dos participantes e fazer o broadcast das mensagens que contém os pedidos de recurso. As chamadas efetuadas aos membros do grupo são realizadas de forma paralela, o que pode vir a melhorar desempenho.

A verificação de exceções não funcionais é um fator importante para o uso deste algoritmo, já que cada host remoto pode ser um ponto de falha. Quando o algoritmo percebe uma exceção não funcional lançada por problemas na comunicação com um host remoto, deixa-se de esperar por sua resposta, o recurso de envio de mensagens é liberado, caso o host que falhou estivesse usando o recurso e o mesmo é retirado do grupo, permitindo a continuidade do algoritmo. A exceção não funcional pode ser vista pelo software IC2D como mostra a Figura 3.

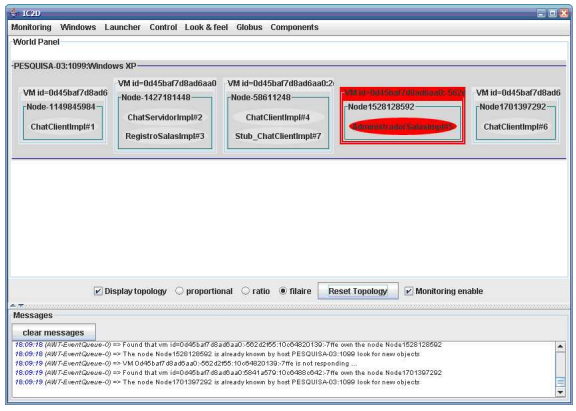


**Figura 3:** Uma exceção não funcional ocorrendo.

Outra característica dos grupos de comunicação no ProActive é a sincronização de chamadas com espera por necessidade, que possibilitou fazer com que um participante só tivesse acesso ao envio de

mensagens assim que todas as chamadas efetuadas ao grupo retornassem. Para evitar problemas com sincronização de relógios, o tempo base para comparação de mensagens é o tempo de chegada no servidor.


**Figura 4:** Um cliente sendo bloqueado.

### 3.3 Algoritmo Token-Ring

O algoritmo *token-ring* utilizou intensamente os recursos do ProActive na sua implementação. O primeiro deles é a classe *Ring*. Graças a essa classe é possível criar uma estrutura de coleção capaz de organizar as referências remotas dos participantes do SynchroTalk em forma anel. Nessa coleção existem métodos para acessar os objetos nas curvas tanto á direita quanto à esquerda do anel, o que torna fácil saber para qual participante o token deve ir, permitindo ainda muda-lo de direção.

Utilizou-se também a tecnologia de agentes móveis disponibilizadas pelo *middleware* para criar um agente que se move pelo anel como se fosse um token. Ao chegar a um host remoto, o agente interage com o participante perguntando se este deseja enviar mensagens para o chat. O agente se move para o próximo host caso o participante não queira usar o recurso ou quando este tiver terminado de enviar mensagens.

Adicionado à utilização de agentes, a aplicação possui a flexibilidade de estabelecer um limite de tempo para a permanência do agente *token* com um participante. Para evitar problemas em caso de falha em algum host, o agente envia uma mensagem informando com qual participante está.

## 4 Avaliação Técnica e Pedagógica

Os algoritmos são avaliados de acordo com dois critérios: aspectos técnicos e aspectos pedagógicos. Ambos estão atrelados, principalmente porque os aspectos técnicos impactam diretamente nos aspectos pedagógicos.

O algoritmo centralizado possui duas vantagens claras: é relativamente simples de ser utilizado, e o coordenador pode a qualquer momento, de acordo com a necessidade, bloquear um ou mais participantes como mostra a Figura 4.

Sob o ponto de vista pedagógico, isso representa algo desejado por parte dos educadores: um ambiente didático onde não haja problemas causados por alunos que conversam paralelamente às aulas ou falem todos ao mesmo tempo, provocando a falta entendimento. Por outro lado, um professor pode impedir um aluno de postar mensagens podendo ceder oportunidade a um aluno que participou menos.

Baseado neste algoritmo, o SynchroTalk abre a possibilidade para que o coordenador seja tanto um humano (professor) quanto um processo computacional. No caso de um humano (professor) ele pode priorizar mensagens de algum(ns) aluno(s), por exemplo, os que estiverem participando de maneira mais ativa e estejam contribuindo com o debate.

O algoritmo centralizado apresenta também uma desvantagem: o coordenador por ser um ponto de convergência de requisições, pode tornar-se um ponto crítico de falhas dependendo da quantidade de participantes. Isso impacta diretamente no processo de ensino-aprendizagem, pois se o número de participantes for relativamente alto o professor pode não conseguir atender a todas as requisições; e de acordo com o fluxo de requisições, o coordenador pode não suportar a carga e perder sua conexão com o servidor deixando a sala de chat sem controle.

No algoritmo *token-ring* os participantes são organizados logicamente em forma de anel e recebem uma permissão (token) que circula entre os participantes. O token é implementado por um agente que pergunta para cada participante se o mesmo deseja falar. O participante pode deixar um botão visual pressionado até que o token chegue na sua máquina. Se o token identificar que o botão está pressionado o aluno recebe automaticamente a permissão para falar.

O algoritmo *token-ring* pode ser um alternativa caso o coordenador assuma uma posição mais passiva, como por exemplo, em atividades onde todos os alunos devem discutir pontos de vistas a respeito de um determinado assunto.

Outras possibilidades de uso do *token-ring* são: se o coordenador precisar se ausentar temporariamente; se a reunião for apenas entre alunos; ou ainda, caso a conexão do coordenador com o servidor seja interrompida.

O *token-ring* também possui algumas desvantagens, pois se num dado momento um aluno que es-

tiver com a permissão (token) perder conexão com a rede, o mesmo não poderá ser movimentado para outros alunos, impedindo assim que os demais recebam permissões para falar. Uma alternativa para esse caso é criar um agente de emergência que circula na direção inversa a do token. A movimentação na direção inversa visa encontrar o token mais rapidamente. Se após uma volta o agente token não for encontrado o agente de emergência cria um novo agente token.

Com relação ao algoritmo distribuído, deve-se mandar a requisição de pedido de palavra para todos os participantes quando se deseja falar. Controlar esse envio não é um processo trivial, pois participantes podem se desconectar a qualquer momento. Em teoria essa possibilidade é uma desvantagem. Entretanto, esse problema foi solucionado com a utilização de grupos de comunicação e com a utilização de exceções não funcionais. Nesse contexto, quando um participante é desconectado os demais deixam de esperar por sua resposta.

Genericamente falando, o algoritmo distribuído funciona de maneira análoga ao algoritmo centralizado, desde que o coordenador seja um processo computacional. Em outras palavras, o primeiro que pedir a palavra a terá. Não obstante, perde-se a possibilidade de dar preferência a grupos de alunos. Isso pode comprometer pedagogicamente o debate se os alunos que conseguiram a palavra não contribuírem significativamente com o debate. Ressalta-se que a contribuição pode vir tanto na forma de afirmações quanto de questionamentos interessantes sobre o tema em discussão.

## 5 Conclusões

Do ponto de vista pedagógico, o sistema de chat proposto vem a contribuir com a educação a distância, visto que se trata de uma ferramenta que pode possibilitar reuniões virtuais eficientes e sem desperdício de tempo com a sincronização dos alunos.

Além de permitir recursos normais de comunicação a distância, a ferramenta possui mecanismo de sincronização que permitem uma melhor organização dos encontros virtuais, já que seus recursos de controle impedem que várias pessoas falem ao mesmo tempo. Nesse contexto, debates, aulas, discussões, entre outros tipos de atividades estudantis poderão ser desenvolvidas com certo grau de segurança caso falte a colaboração do(s) aluno(s). Sendo assim, o professor terá formas de garantir a organização do espaço virtual.

Não obstante, com o *SynchroTalk* abre-se espaço para discussões acerca de como a educação a distância mostra caminhos para a expansão do ensino. No entanto, é necessário pensar além de números. A qualidade do ensino é um fator fundamental, pois a educação a distância deve visar muito mais que comunicação entre indivíduos geograficamente distantes, mas também prover maneiras de comunicação organizada que atenda às necessidades de professores e alunos.

Contemplando o ponto de vista tecnológico, a ferramenta oferece a possibilidade de funcionamento em qualquer sistema operacional. E, possibilita aos envolvidos neste trabalho experiências nas novas tecnologias disponíveis.

## 6 Trabalhos Futuros

Inicialmente, pretende-se testar o SynchroTalk por meio de aulas a distância em diversas disciplinas para verificar a eficiência das soluções.

Devido às vulnerabilidades que cada algoritmo pode apresentar e ainda pelo fato de aplicações distribuídas estarem sujeitas à problemas na comunicação em rede, pretende-se explorar os recursos de tolerâncias à falhas disponibilizados pelo ProActive.

No mecanismo centralizado espera-se conseguir eleger um host para coordenar o envio de mensagens, em caso de problemas com o coordenador anterior. Ainda com os recursos de tolerância à falhas, espera-se conseguir gerar um agente móvel inteligente, que controle o tráfego de mensagens na rede.

Espera-se encontrar formas de minimizar também os pontos de falha do mecanismo distribuído.

**Agradecimentos**

Os autores gostariam de agradecer o apoio financeiro do CNPq sem o qual este trabalho não seria realizado.

## Referências

[1] BADUEL, L., Baude, F., and Caromel, D. Efficient, flexible, and typed group communications in java. In *Joint ACM Java Grand - ISCOPE 2002 Conference*, pages 28–36, ACM Press, 2002.

[2] CORBA. Corba's home page. http://www.corba.org, Visitado em 02/04/2006.

[3] COULOURIS, G., DOLLIMORE, J., and KINDBERG, T. *Distributed Sys-tems: Concepts and Design*. Adson Wesley, 3 edition, 2000.

[4] LITWIN, E. *Educação a Distância: Temas para o Debate de uma Nova Agenda Educativa*. Artmed, Porto Alegre-RS, 2001.

[5] Microsystems, S. Sun's home page. http://www.sun.com, Visitado em 02/04/2006.

[6] OBJECTWEB. Proactive. http://www.objectweb.org/ProActive, Visitado em 25/05/2006.

[7] OEIRAS, J. Y., LACHI, R. L., and da ROCHA, H. V. Uma ferramenta de bate-papo com mecanismos de coordenação para apoio a discussões online. In *SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*, Manaus-AM, 2004.

[8] OEIRAS, J. Y. Y. and ROCHA, H. V. Uma modalidade de comunicação mediada por computador e suas várias interfaces. In *Workshop Sobre Fatores Humanos em Sistemas Computacionais*, pages 151–160, UFRGS - Porto Alegre - RS, 2000.

[9] PIMENTEL, M. G. and SAMPAIO, F. F. Hiperdiálogo uma ferramenta de bate-papo para diminuir a perda de co-texto. In *SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*, pages 21–23, Vitória-ES, 2002.

[10] SMITH, M., CADIZ, J. J., and BURKHALTER, B. Conversation trees and threaded chats. *SIGCHI Bulletin*, 3(31):21–23, 1999.

[11] SPACE, O. http://www.objectspace.com, Visitado em 02/04/2006.

[12] STAFFORD, T. F. Understanding Motivation for Internet Use in Distance Education. *IEEE Transaction on Education*, 48(1):301–306, 2005.

[13] TANENBAUM, A. S. *Distributed Operating Systems*. Prentice Hall, New Jersey, 1995.

[14] TANENBAUM, A. S. *Sistemas Operacionais Modernos*. Prentice Hall, São Paulo, 2003.

[15] VAHL JÚNIOR, J. C. Uso de agentes de interface para adequação de bate-papos ao contexto de educação a distância. Master's thesis, Instituto de Computação, Universidade Estadual de Campinas - UNICAMP, Campinas-SP, 2003.