PROJETO E IMPLEMENTAÇÃO DE UM AMBIENTE DE ESCRITA COLABORATIVA BASEADO EM UMA PLATAFORMA DE SUPORTE PARA DISTRIBUIÇÃO

Marcelo D. Vessoni [1]
e-mail: vessoni@dc.ufscar.br

Luis C. Trevelin [2]
e-mail: trevelin@power.ufscar.br

Universidade Federal de São Carlos
Programa de Pós Graduação em Ciência da Computação
Rodovia Washington Luiz, Km 235 - Caixa Postal 676
13.565-905 - São Carlos - SP - FAX/Fone: 016 260 8233

Resumo

Este artigo apresenta uma proposta de um ambiente integrado de edição de documentos compartilhados projetado a partir do levantamento de um número significativo de aspectos de design importantes para a construção de ferramentas para escrita em grupo. Serão discutidos aspectos de implementação referentes a detalhes da arquitetura proposta e da utilização de uma plataforma de suporte composta de *frameworks* para desenvolvimento de aplicações multimídia distribuídas. Estas aplicações para execução em ambientes Intranet/Internet são construídas através da linguagem Java com integração aos serviços oferecidos pela plataforma em questão.

Palavras Chave: Aplicações Distribuídas e Trabalho Cooperativo, Groupware, Escrita Colaborativa

Abstract

It is presented in this paper a proposal of an integrated environment for shared documents editing, projected after a survey of a considerable number of design issues for building up group work tools, as a synchronous shared text editor. It is discussed some implementation aspects relative to the proposed architecture and the use of a support platform composed of frameworks for building distributed multimedia applications. These applications, designed to run into Intranet/Internet environments, are built using the Java programming Language integrated with the services given by that platform.

Keywords: Distributed Applications and Cooperative Work, Groupware, Collaborative Writing

[1] Financiado por CNPq
[2] Financiado por FINEP/RECOPE - Projeto MultiEng

1. Introdução

A disciplina CSCW (C*omputer-Supported Cooperative Work*) surgiu em meados dos anos 80 quando o crescente número de computadores conectados por uma rede de comunicação mudou o conceito de interação homem-computador [16]. Ao invés de continuar o desenvolvimento de aplicações que permitiam apenas o trabalho individual, pesquisadores e desenvolvedores definiram a possibilidade das pessoas trabalharem em conjunto por meio de aplicações construídas para esta finalidade. A partir do desenvolvimento desta disciplina, obteu-se um grande número de entusiastas no estudo de como as pessoas trabalham em conjunto e como o computador e tecnologias relacionadas afetam o comportamento de um grupo.

Como populares aplicações mono-usuárias, os editores de texto se tornaram um alvo natural para a inserção em um contexto de funcionamento em grupo, ou seja, se modificando para se tornarem uma aplicação de groupware, por vários motivos. Entre eles, sistemas para escrita colaborativa poderiam beneficiar muitos usuários, onde seriam utilizados como um valioso meio para as pessoas compartilharem o trabalho entre elas de forma imediata. Além disso, estes sistemas de groupware permitem o avanço de pesquisa em diversas áreas devido aos seus problemas técnicos inerentes, como os aspectos relativos a sistemas distribuídos, multimídia, comunicação e desenvolvimento de toolkits.

Neste artigo é apresentado o projeto de um ambiente de groupware para co-autoria de documentos texto a partir do levantamento de requisitos necessários a caracterização de sistemas para escrita colaborativa síncronos. Discutem-se também aspectos de implementação, onde são utilizados serviços fornecidos por uma plataforma específica para o desenvolvimento de sistemas distribuídos tanto para ambientes Web como para aplicações multiplataformas em Java. Essa plataforma, nomeada JAMP (Java Architecture for Media Processing) [7], a ser apresentada em seção posterior, é composta por um conjunto de *frameworks*, sendo responsável, por exemplo, por iniciar um processo de *Trading* com a utilização de um *broker* totalmente desenvolvido em linguagem Java, o JavaBroker. Juntamente com esta plataforma, apresenta-se a proposta e implementação de uma arquitetura cliente/servidor integrada ao *middleware* JAMP do ambiente de edição textual colaborativa denominado CoopWrite. A visão geral do ambiente proposto é mostrado na figura 1.

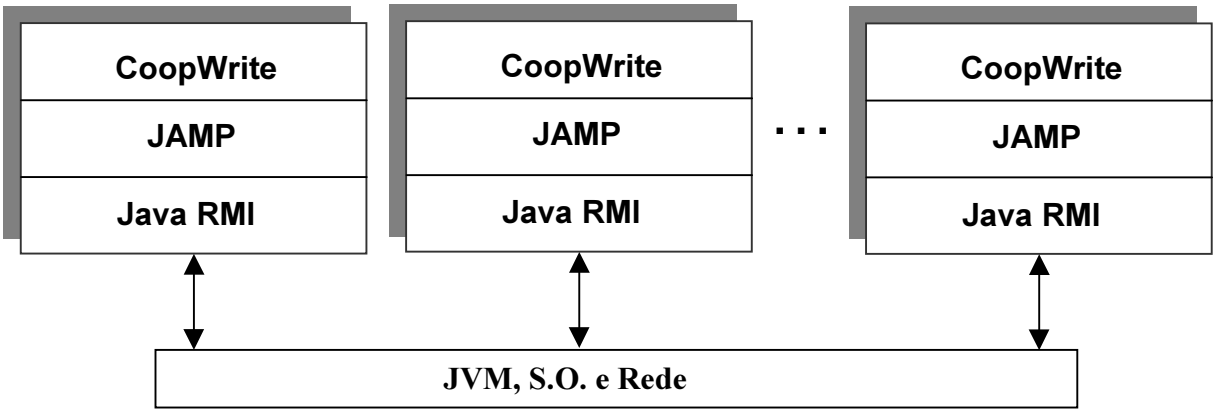


Figura 1 – Visão geral do ambiente CoopWrite

2. Escrita Colaborativa : Requerimentos de Projeto

Quando uma aplicação de Groupware é implementada como um sistema distribuído, vários aspectos de projeto devem ser considerados como críticos para o sucesso da aplicação. Para prover suporte computadorizado para a escrita colaborativa síncrona, há um número de requerimentos que o projeto de um editor de texto colaborativo deve satisfazer [18]. Aspectos importantes no projeto de um ambiente para escrita colaborativa incluem suporte para um artefato compartilhado, onde levam-se em consideração os mecanismos de controle de concorrência e como estes mecanismos afetam as interações [9]; consciência (awareness) das atividades e localização dos colaboradores dentro do espaço de trabalho compartilhado; e comunicação dentro do espaço de tarefas compartilhadas fornecido pelo sistema [13].

Editores de texto compartilhados fazem parte de uma classe de sistemas de groupware que suportam espaços de trabalho compartilhados altamente interativos e em tempo real [1].

Espera-se que os participantes destas conferências:

- Estejam em comunicação em tempo real uns com os outros, através de canais de conversação de texto, áudio e vídeo.
- Controlem uma janela de edição comum, que consiste em um espaço de trabalho visual compartilhado pelo grupo que mantém instâncias distintas do editor, no qual se realiza o trabalho de edição colaborativa.
- Estejam cientes de cada ação (com qualquer granularidade) dos outros participantes dentro do espaço de trabalho.
- Possam utilizar opções de requisição para entrar em uma sessão já em ocorrência, pedido de saída de uma sessão e listar todas as conferências e seus grupos .
- Possam alternar entre modos de edição colaborativa ou individual (edição privada)

As discussões acima sugerem que a introdução de ferramentas de computadores para suporte à escrita colaborativa terá um impacto significativo na comunicação que ocorre durante o processo de escrita conjunta. Escrita colaborativa envolve muitas pessoas trabalhando juntas, em lugares distintos ou não e também em tempos iguais e diferentes. Sistemas projetados para suportar a escrita colaborativa tem sido tradicionalmente divididos em duas categorias: a escrita dos diferentes autores é síncrona, acontecendo ao mesmo tempo, ou assíncrona, acontecendo em tempos diferentes [4]. Esta divisão não é absoluta, dado que alguns sistemas semi-síncronos também foram desenvolvidos. Estes diferentes modelos de escrita colaborativa requerem suporte computadorizado muito distintos. Exemplos de sistemas para escrita colaborativa síncrona são os sistemas GROVE [4,5] e o ShrEdit [12,15].

2.1 Controle de Concorrência

Problemas de controle de concorrência aparecem quando um programa de computador, dados ou interfaces estão distribuídos sobre muitos computadores ligados em rede.

O fato dos usuários poderem realizar operações em objetos compartilhados que podem ser executados em ordens distintas e em diferentes computadores introduz a probabilidade de conflito. Este conflito pode destruir o funcionamento do espaço de trabalho compartilhado, levando a uma confusão no grupo e uma quebra na colaboração.

Um sistema de escrita colaborativa provê um espaço de trabalho texto compartilhado, e com isto necessita manter consistente a visão deste espaço por parte de todos os outros usuários durante o acesso concorrente ao texto. Para isto, é necessário algum mecanismo de controle de concorrência como forma de coordenação do trabalho colaborativo. A maior dificuldade em se implementar um editor de texto compartilhado síncrono reside na granularidade do mecanismo de concorrência, sendo que esta granularidade do controle requerido pode variar através do processo de escrita.

Para manter a coordenação da difusão das atualizações concorrentes em uma forma transparente ao usuário, ou seja, sem que ocorra atrasos na cadência do seu trabalho, é necessário que o mecanismo de concorrência ofereça tempos de respostas aceitáveis [8].

2.1.1 Tipos de Controle de Concorrência

Em sistemas de groupware da classe de ferramentas de co-autoria, as ações no espaço compartilhado são potencialmente conflitantes, sendo necessária alguma forma de controle. Uma apresentação detalhada dos compromissos envolvidos entre a interface e o uso das diferentes abordagens para controle de concorrência em groupware em tempo-real é dado por Greenberg e Marwood [9]. A seguir são abordados os principais tipos de controle de concorrência para este tipo de aplicação.

Sem controle de concorrência. O controle de concorrência pode seguramente ser ignorado em sistemas onde o nível de interação e a granularidade das mudanças em objetos atinge níveis de inconsistências não visíveis. Esta abordagem não funciona em editores de texto, onde diferenças muito visíveis ocorrem.

Serialização. A serialização pode ser mantida tanto sincronizando eventos entrelaçados de tal forma que transações atômicas são executadas serialmente através de todo o sistema ou então reparando os efeitos dos eventos fora de ordem para dar a impressão que eles são executados serialmente. Serialização *não-otimista* assegura que eventos são sempre executadas na ordem correta em todos os sites através da espera em receber um evento prévio antes da execução de um dado evento, o que pode reduzir a performance. Serialização *otimista* é baseada na hipótese de que eventos conflitantes raramente são recebidos fora da ordem, e que é mais eficiente executar os eventos como foram recebidos e ocasionalmente reparar problemas.

Ellis e Gibbs [6] argumentam que é questionável se a serialização completa é sempre apropriada para aplicações colaborativas interativas. A aceitação da serialização também depende da granularidade das interações. Quando trabalhando com granularidade fina, os

usuários tendem a não notar os efeitos inesperados da serialização, enquanto com granularidade mais grossa, o ato de refazer ou desfazer ações pode se tornar inconsistente.

Travamento (*Locking*). Esta outra abordagem para controle de concorrência dá a um processo o acesso exclusivo a um objeto compartilhado. As travas são requisitadas quanto um processo quer acessar um recurso. Se ninguém mais possui a trava, o recurso é garantido, caso contrário é negado. Quando a trava requer que o site requisitante espere até que a trava seja liberada antes de agir no recurso, esta abordagem é denominada *não-otimista.* Com travas o*timistas*, é permitido ao site tentar um acesso à trava, recebendo assim uma *aprovação de tentativa*, e começar a manipular o objeto antes de saber se realmente possui a trava. Se a trava é aprovada, o acesso continua normal, caso contrário, o objeto deve voltar ao seu estado original.

O modo pelo qual os usuários são forçados a esperar pelas travas tem um impacto na interface. Com travas *não-otimistas*, os usuários têm que esperar pela trava, e há chances de não conseguir acessá-las em tempo hábil. Isto pode ser inaceitável se estes atrasos são percebidos pelos usuários, pois interferirão no fluxo de interação, fazendo o sistema parecer lento, principalmente quando compondo um texto.

2.1.2 Implementação de Concorrência em Groupware Síncrono

Os métodos para garantir consistência em espaços de trabalho texto compartilhados devem apresentar um compromisso entre a liberdade da escolha da granularidade do controle pelo usuário e os tempos de resposta para a obtenção do recurso (no caso, fragmento do texto).

Controle Flexível – Escolha da Granularidade. Uma solução aceitável para tornar flexível e consistente um mecanismo de concorrência com implementação algorítmica mais simples seria uma combinação de serialização e travamento. As travas seriam disponíveis através dos diversos níveis do documento, como palavra, linha, parágrafo e uma seleção do texto feita pelo usuário, sendo que este faria a escolha do nível de trava através da interface do sistema. Uma sugestão feita por Mitchell [13], seria a implementação das travas através de uma combinação de um servidor de comunicação central e travas replicadas em cada site cliente. Além disso, uma mistura de diferentes níveis de travas poderia ser usada dentro de cada sessão ativa. O sistema local determina o nível da trava a ser usada localmente, e interpreta novas requisições. Isto permite que cada cópia local da conferência texto tenha um nível diferente de controle de acesso.

Consistência. Para assegurar consistência, quando uma requisição para travamento é feita, a requisição pode ser serializada através do servidor central. Isto assegura que todas as requisições por travas sejam vistas na mesma ordem em todas as cópias da conferência. Quando a requisição for recebida, o usuário local verifica se a região requisitada está livre, e em caso positivo, a trava é garantida. Quando uma inserção no texto é realizada, ela é primeiro executada localmente e depois enviada para as outras cópias da conferência através do servidor de serialização.

2.2 Consciência (*Awareness*)

A consciência da presença e ações dos participantes da conferência pode ser obtida através de diversas maneiras diferentes. Estas abordagens podem ser divididas em ferramentas que provêem consciência periférica, e no suporte a um senso de histórico e contexto do andamento da colaboração e do desenvolvimento do texto compartilhado.

Consciência Periférica. Consciência periférica envolve manter todos os usuários cientes do fato de que eles estão trabalhando em um espaço de trabalho compartilhado e prover uma notificação não-intrusiva das ações e localização de todos os outros usuários. Exemplo disso é o uso de cores diferentes para cada usuário que entrou no sistema de modo a identificá-lo. O sistema também poderia manter informações disponíveis sobre o restante do grupo e o estado do texto compartilhado, fornecendo um contexto compartilhado para encorajar a comunicação e cooperação.

Esta discussão mostra a importância de manter um consciência tanto do espaço compartilhado onde se está trabalhando quanto do fato de que há outras pessoas trabalhando naquele espaço. Dourish e Belloti [3] indica que o uso de um *feedback compartilhado*, ou seja, a noção de prover uma informação periférica implícita sobre todos os usuários no espaço compartilhado, é uma abordagem promissora para prover consciência mútua. Exemplo desta abordagem pode ser visto no sistema ShrEdit [12,15], o qual permite ao usuário requisitar que o sistema procure e rastreie os movimentos (em termos de atividades realizadas no mesmo texto compartilhado) dos outros usuários, mas não provê *feedback* contínuo aos usuários para indicar a localização das atividades de edição dos outros participantes. Ao mesmo tempo, usuários precisam ser capazes de explicitamente requisitar informações sobre o contexto e o conteúdo do espaço de trabalho compartilhado, incluindo informações sobre quem fez contribuições e mudanças no documento e quando.

Ao prover consciência periférica da presença do usuário local e dos outros usuários dentro do espaço de trabalho compartilhado, o objetivo deveria ser tornar óbvio aos usuários a natureza compartilhada do espaço de trabalho e encorajar a comunicação entre os colaboradores. Consciência periférica inclui ciência das ações e potenciais ações dentro do espaço de trabalho[13]. O sistema deveria fornecer informações sobre quem fez as mudanças no texto, quando e porquê, além de informar sobre o que os outros usuários estão fazendo e como estas ações afetam o usuário local e vice versa. Deve-se tornar óbvio quem está fazendo e quem fez algo, tornando visíveis ao grupo os eventuais conflitos.

Visão Radar. Os sistemas de groupware mais recentes têm relaxado o modelo WYSISWIS "What you See Is What I See", onde todos os participantes têm a mesma visão da área de trabalho compartilhada durante toda a conferência [10]. Este relaxamento dá aos usuários a possibilidade de controlar suas próprias visões dentro do espaço de trabalho, permitindo a eles trabalharem em um estilo mais natural. Contudo, este relaxamento pode contribuir para a perda da ciência de que se está trabalhando com um grupo, pois ao se preocupar com apenas a sua visão, o usuário pode perder o controle de onde os outros usuários estão e o que eles estão fazendo no espaço de trabalho.

Uma alternativa para manter a consciência do compartilhamento seria o uso de uma janela auxiliar que daria um overview geral sobre o documento, apresentando uma miniatura do espaço de trabalho e indicando a localização exata de cada participante da conferência. Essa técnica é conhecida como Visão Radar [2].

Histórico de Mudanças. Outra característica importante de ser apresentada por uma ferramenta colaborativa é a possibilidade de manter um histórico sobre as mudanças em um documento compartilhado. Esta informação deveria ser armazenada com o espaço de trabalho compartilhado, a qual mostra o desenvolvimento do documento e a conexão entre este histórico e a comunicação que ocorre ao redor do texto. Uma alternativa para prover a informação necessária sobre mudanças no documento é através da visualização das diferenças no documento, técnica conhecida como "*diffs*" [14]. Para que um usuário possa saber, por exemplo, quem fez determinada alteração ou quando foi a última vez que determinado parágrafo foi editado, é necessário que o sistema possa manter alguma forma de registro das mudanças no texto, como informações sobre o "proprietário" da ação e seu acesso. Isto pode ser feito novamente utilizando cores como indexador para os usuários.

2.3 Espaços de Trabalho Alternativos e Comunicação Inter-Usuários

Uma característica importante é a possibilidade de refinar e aperfeiçoar as idéias antes de transmiti-las aos outros participantes da conferência distribuída. Isto sugere que o editor colaborativo deveria permitir uma edição privada de partes do texto sendo compartilhado, possibilitando sempre uma movimentação livre entre o espaço de edição público e o espaço de edição privado.

Janela de Texto Privada. Esta janela de texto privada seria apresentada ao usuário no momento em que ele necessitasse refinar um fragmento de texto recém-construído antes de torná-lo compartilhado para visualização pelo grupo. Esta janela poderia ser iniciada a partir da seleção do fragmento de texto e chamando a janela privada a partir de uma opção em algum menu. Assim, o texto poderia ser editado livremente com privacidade.

Em termos de requerimentos de consciência, o resto do grupo poderia manter uma ciência do fato de que um usuário estaria trabalhando em um espaço privado, pois encorajaria a comunicação sobre o texto e reduziria a possibilidade de mudanças conflitantes no texto. Ellis, Gibbs e Rein [5] sugerem utilizar uma forma de indicar a região onde um usuário remoto está fazendo alguma mudança privativamente, por exemplo, tornando a região sublinhada por pontos. No momento da edição privada, este pontilhamento é propagado para todos, e se a mudança é cancelada, o texto é revertido para sua aparência normal.

Canal de Comunicação Inter-Usuários. O principal canal de comunicação indireta entre os participantes da conferência em um editor colaborativo é o espaço de trabalho texto compartilhado. Isto provê um contexto compartilhado para o grupo, um espaço para a realização concreta das idéias. Entretanto, o uso apenas o documento texto compartilhado como canal de comunicação indireta entre os usuários limita sua comunicação mais informal. Esta limitação aparece no momento de necessidade de discussão ou explicação sobre algum

trecho inserido no texto por parte de algum participante, ou até mesmo como forma de negociação prévia sobre a construção do documento como um todo.

Para facilitar esta necessidade de comunicação mais direta, um espaço separado para uma conversação informal poderia ser implementado. Esta espécie de Chat embutido no editor colaborativo funcionaria da forma usual: envio de mensagens texto para determinado usuário através de seu identificador (nome, por exemplo), ou para todo o grupo, facilitando a discussão de idéias em conferências de edição colaborativa totalmente distribuídas.

3. JAMP – Um Suporte para Desenvolvimento de Aplicações Distribuídas

A plataforma JAMP (Java Architecture for Media Processing) apresentada em [7] é uma plataforma para desenvolvimento de aplicações multimídia em ambientes distribuídos abertos. Ela consiste de um conjunto escalável de servidores e frameworks utilizados pela aplicação multimídia. A funcionalidade de cada framework é incorporada pela aplicação através do uso dos métodos do framework e posterior recompilação, habilitando assim a execução dos serviços multimídia oferecidos. A figura 2 apresenta a visão geral da plataforma JAMP.

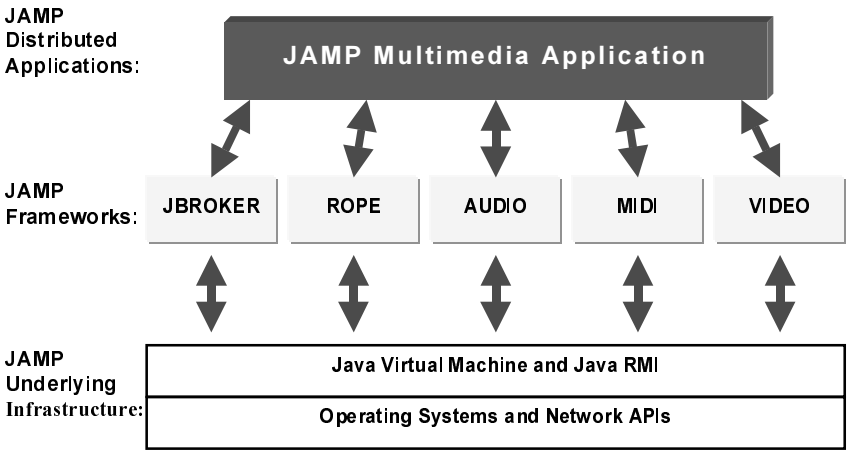


Figura 2 – Plataforma JAMP e seus frameworks [7]

Embora o framework de chamada remota de métodos da linguagem Java, RMI (Remote Method Invocation), integre o ambiente Java ao modelo computacional de objetos distribuídos, ele não é suficiente para prover um ambiente aberto flexível para processamento distribuído. Um sistema distribuído contém objetos compartilhados que executam em localidades específicas. Porém, para o resto do sistema, esta localização deve ser transparente. Um broker é capaz de providenciar esta transparência, além de adicionar flexibilidade e portabilidade para objetos em um ambiente distribuído.

O interesse maior na arquitetura JAMP reside no seu principal framework, o *Jbroker*, que oferece as facilidades mostradas acima. Este é o framework *middleware*, um broker nomeado *JavaBroker* que contém uma base de dados dos servidores disponíveis no momento,

habilitando os clientes a localizarem os servidores distribuídos (objetos remotos) através da rede. O uso dos serviços de um servidor ocorre por RMI, após a referência do mesmo ter sido obtido com sucesso. Este processo, que inicia com o servidor sendo registrado pelo broker e a aplicação obtendo a sua referência é chamado de Trading.

Para a utilização da plataforma JAMP como suporte à aplicação colaborativa, faz-se necessária a definição de uma arquitetura cliente/servidor com os seu módulos escritos em linguagem Java. Os componentes cliente e servidor são escritos incorporando os serviços e frameworks oferecidos pela plataforma.

Comunicação em grupo. Os frameworks JAMP são extensíveis, podendo ser escritos outros que apresentem novos recursos conforme as necessidades dos desenvolvedores. Uma vez que uma das principais vantagens para a utilização de frameworks é a possibilidade de reutilização de código, um dos serviços que justificam a utilização de re-uso em um sistema colaborativo é o serviço que ofereça facilidades para comunicação em grupo, tais como:

- O uso de endereçamento em grupo, através de mecanismos de multicasting. Isto elimina do escopo do desenvolvedor de se preocupar com detalhes de endereçamento dos processos individuais que compõem o grupo.

- A liberação única das mensagens para o grupo (sem perda ou duplicação, garantindo a tolerância a falhas)

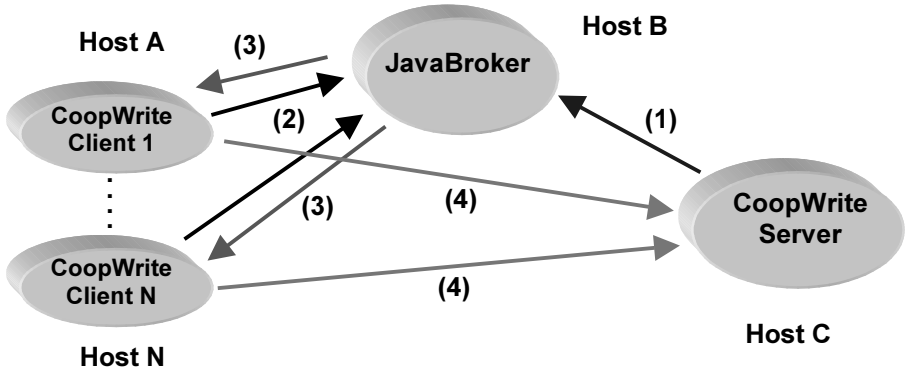
Este framework para suporte aos mecanismos de endereçamento em grupo (Multicasting) está sendo desenvolvido pelo grupo de pesquisa e será incorporado ao ambiente de escrita colaborativa CoopWrite para utilização das facilidades oferecidas pela real comunicação em grupo.

4. CoopWrite – Aspectos de Arquitetura

O ambiente de escrita colaborativa CoopWrite foi proposto como forma de experimentar os requisitos levantados como sendo importantes para o sucesso de uma aplicação de groupware que exija funcionamento síncrono e que satisfaça as definições de sistemas de co-autoria. Para isto, o sistema deve apresentar módulos que gerenciem o mecanismo de concorrência empregado, implementação do conceito de consciência (awareness) das atividades dos usuários (através de cores diferentes para identificação do usuário e ferramentas auxiliares, como a visão Radar) e modos de comunicação inter-usuários separados do espaço de trabalho compartilhado.

Este ambiente de Escrita Colaborativa faz parte de um conjunto de aplicações para trabalho em grupo inseridos no contexto de um projeto de desenvolvimento destas aplicações em pesquisa no programa de pós graduação (MultiEng). Estas aplicações, um ambiente de Suporte a Reuniões Eletrônicas e Apoio à Tomada de Decisão (SACE-CSCW) [17] e um ambiente de Edição Gráfica Colaborativa, utilizam a plataforma JAMP como middleware para distribuição.

Uso do Middleware. Proposta como uma arquitetura distribuída cliente/servidor, há a utilização do *JavaBroker* permitindo ligação sob-demanda (on-demand) entre a aplicação distribuída e seu servidor, cuja localização é transparente ao cliente. Na codificação da aplicação, é necessário apenas informar o nome do servidor a ser registrado, não se importando com sua localização física em termos de endereçamento e host. Durante a codificação das interfaces que contém as definições do métodos remotos para comunicação RMI, há a utilização da interface *Tradeable* do JavaBroker ao invés da interface padrão *java.rmi.Remote* da linguagem Java (esta interface por sua vez é utilizada pela interface *Tradeable*). A figura 3 apresenta a integração entre o sistema e a plataforma de suporte, mostrando o processo de tranding.



(1) - Servidor exporta referência ao JavaBroker

(2) - Cliente procura por serviço

(3) - Cliente obtém referência do servidor

(4) - Cliente em comunicação direta (RMI) com servidor

Figura 3 – Visão do processo Trading

No componente servidor, há a exportação da referência do servidor ao JavaBroker através do código exemplificado abaixo:

```
  try {
    session.broker.export( "CoopWrite Server", new CoopWriteServer() );
    . . .  }
  catch( RemoteException e ) {…}
```

A invocação da referência do servidor pelo componente cliente é obtida através do seguinte trecho de código:

```
try {
obj = (CoopWriteServerInt) session.broker.invoke( "CoopWrite Server" );
. . .
```

É importante ressaltar que o código de gerenciamento da comunicação RMI é totalmente transferida para o código do JavaBroker, onde ocorre a instalação do gerenciador

de segurança do RMI (*System.setSecurityManager(...)*) e o *rebind* do objeto para um dado nome no *rmiregistry*.

Módulos do Sistema. A arquitetura do ambiente CoopWrite é proposta como um conjunto de classes (módulos) responsáveis por funções específicas dentro do sistema, uma vez que a linguagem Java permite apenas a utilização da orientação a objetos.

Um dos módulos mais importantes do componente cliente é o que cria o espaço de trabalho compartilhado, ou seja, a área em que o documento é composto. Visando eliminar o desenvolvimento completo de um editor para a edição do documento, a forma encontrada de minimizar este desenvolvimento adicional é utilizar um editor projetado para uso mono-usuário e, com alterações no código, modificar a área de edição do documento para torná-la um espaço de trabalho compartilhado. Além disso, é necessária a integração deste editor com os módulos responsáveis pela caracterização de um ambiente para escrita colaborativa.

A aplicação mono-usuária escolhida é um editor estilo StylePad, incluso como amostra da biblioteca de interface de usuário gráfica do *Java Foundation Classes*, conhecida como *Swing* [11] e também fornecida com o kit de desenvolvimento Java da Sun, JDK 2. A figura 4 apresenta a interface do StylePad.

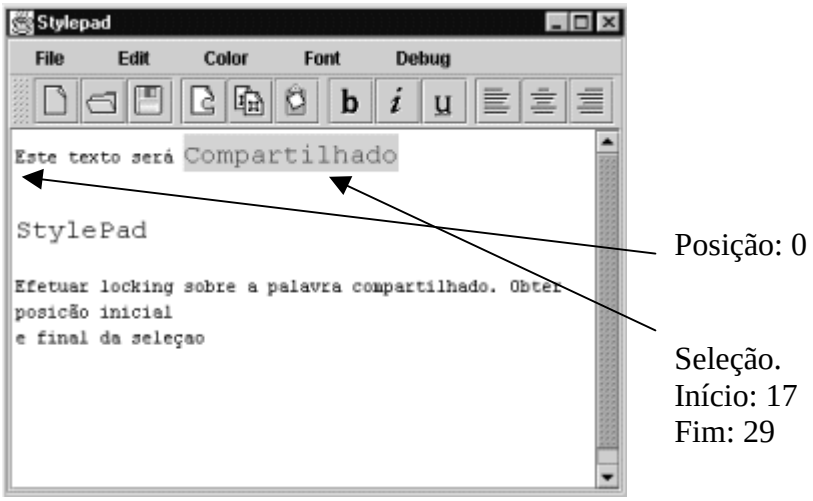


Figura 4 – StylePad – editor de texto [11] e sistema de coordenadas empregado

Esta aplicação contém um painel rolável que incorpora um componente de edição de texto, o *JTextPane*. O pacote de gerenciamento de texto do Swing Java facilita a implementação de edição concorrente do texto, pois apresenta um modelo chamado *document* (e uma sub-interface *StyledDocument*) que gerencia o conteúdo do componente. Este gerenciamento permite, entre outras funcionalidades, extrair o conteúdo do documento, notificar eventos com a posição corrente do cursor ao inserir ou remover fragmentos, e realizar seleções de texto com obtenção das posições de início e fim.

As funcionalidades dos módulos que compõem o sistema serão apresentados a seguir. A visão geral da arquitetura CoopWrite proposta é mostrada na figura 5.

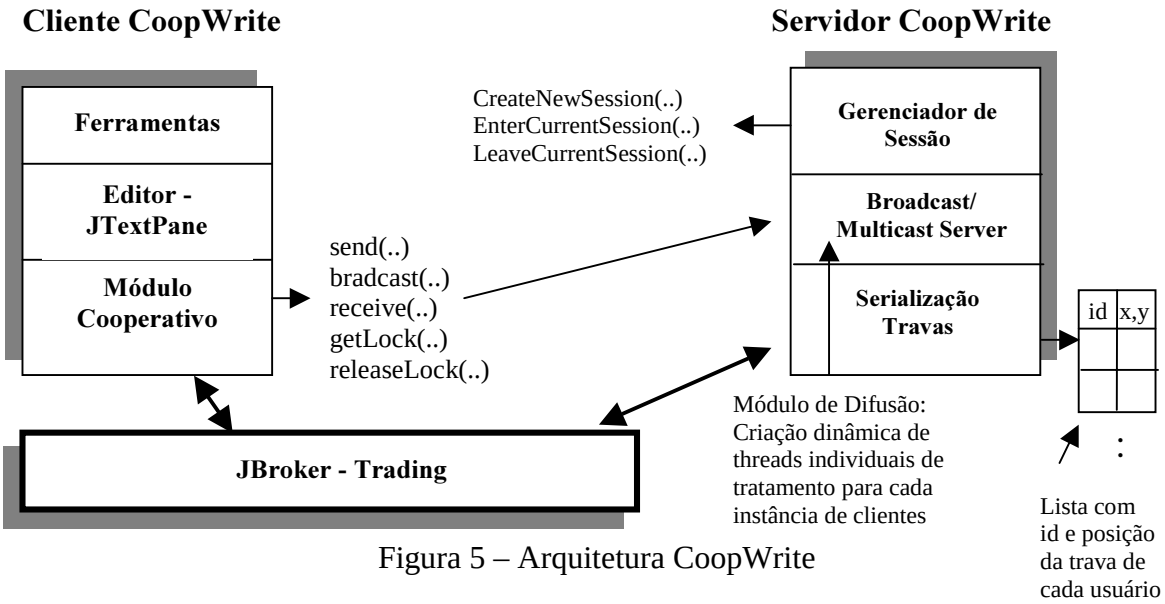
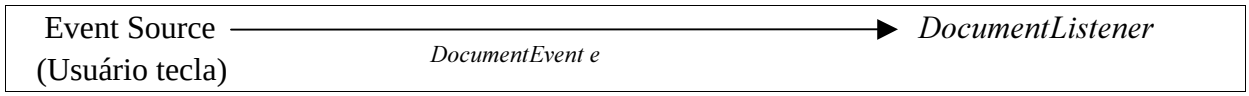


Figura 5 – Arquitetura CoopWrite

Módulo Cooperativo (Cliente). Este é o módulo responsável pelo gerenciamento do documento em edição local no componente *JTextPane* do editor e também pelo tratamento do controle de concorrência. Responsável por:

- extrair o conteúdo do componente de texto e empacotar mensagens de notificação de mudanças no texto (ex.: inserção e remoção) para serem enviadas ao servidor e então distribuídas ao grupo. Trata também a chegada das mesmas mensagens remotas e realiza a alteração necessária no documento local. As mudanças que ocorrem no texto são interceptadas por um *DocumentListener*, que captura os eventos de inserção, remoção e mudança no estilo, além de permitir a sobrecarga dos métodos que executam estas notificações.



Quando o usuário tecla no teclado, um evento de documento é gerado e capturado pelo *Listener* apropriado

- ativar o módulo de gerenciamento de sessão do servidor no momento da instanciação de um novo participante da conferência. Esta entrada no sistema requer o preenchimento de uma tela de login, onde o usuário fornece seu nome para identificação.

- tratar uma requisição de travamento (locking), interceptando o evento de pedido de travamento feito pela interface (empacota em uma mensagem com as coordenadas do fragmento do documento) e a envia ao servidor para ser difundida. Também espera pela mensagem de permissão ou não (em caso de conflito) do uso do token de trava serializado pelo servidor.

- recebe e executa as requisições de travamento serializadas pelos outros usuários e que foram difundidas pelo servidor.

- trata o comando de liberação da trava feito pelo usuário local.

Módulo de Serialização de Travas (Servidor). Módulo responsável por tratar as mensagens de requisição por travamento (locking) enviadas pelos clientes.

- serializa as requisições através de rotulamento, assegurando que todas as requisições por travas sejam executadas na mesma ordem em todos os clientes, e as enviam aos clientes para serem tratadas.

- mantém um registro (uma lista encadeada) contendo informações sobre todas as travas concedidas até o momento, incluindo o identificador do proprietário da ação e as coordenadas relativas ao fragmento do documento bloqueado. Através da varredura deste registro, os pedidos de travamento são concedidos ou negados ao módulo cooperativo do cliente.

Módulo Gerenciador de Sessão. Módulo responsável pelo registro da identificação dos usuários 'logados' no sistema e determinação da cor que será atribuída a cada participante. Implementa os seguintes módulos remotos:

- CreateNewSession( ) – Participante da conferência solicita criação de um novo grupo de trabalho, tornando a área de trabalho disponível para a criação de um novo documento.

- EnterCurrentSession( ) – Usuário obtém listagem das conferências correntes no momento e se torna participante da escolhida. O host cliente executa carregamento do documento no espaço de trabalho, o qual está presente como uma cópia da conferência no servidor, para salvamento em disco.

- LeaveCurrentSession( ) – O usuário deixa a conferência corrente, com o grupo recebendo notificação da ação. Com essa ação, todas as travas pendentes sobre o documento feita pelo usuário são liberadas.

Ferramentas (Tools). Consiste de um conjunto de ferramentas que visam dar suporte ao usuário e ao grupo durante a conferência, implementando algumas características funcionais apresentadas anteriormente.

    - **Chat.** Ferramenta que implementa a comunicação inter-usuários, permitindo o envio de mensagens texto para todo o grupo ou para um usuário isolado e desta forma sendo útil para a discussão de idéias.

    - **Janela Privada.** Janela para edição privada de um fragmento do documento.

    - **Visão Radar.** Mecanismo em implementação que visa apresentar uma visão em miniatura do documento, permitindo a visualização da posição de cada participante da conferência (awareness detalhado).

Painel de Controle do Ambiente CoopWrite. Consiste em uma janela suspensa durante a conferência onde o participante tem acesso às funcionalidades do sistema. Após a tela de login, essa é a janela principal de controle. É apresentada na figura 6.

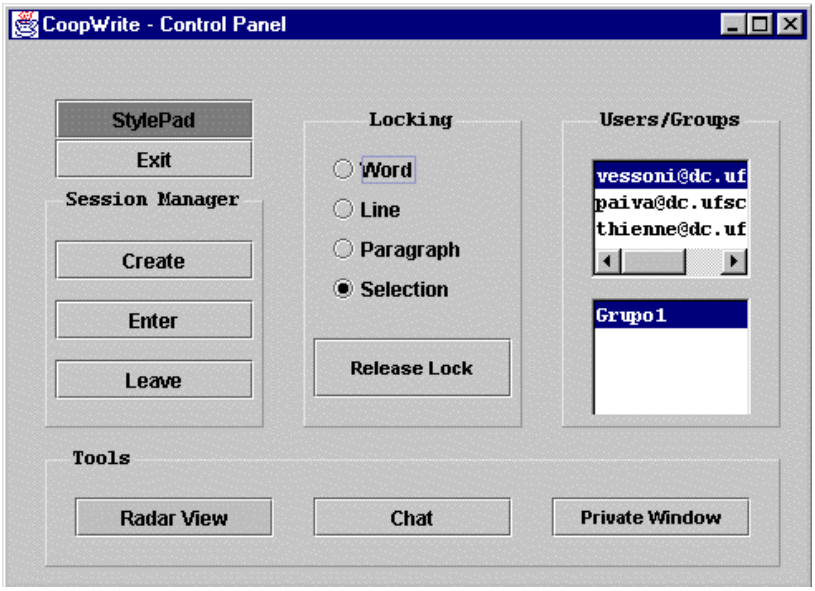


Figura 6 - Painel de Controle do Ambiente CoopWrite

O painel de controle possui acesso ao módulo de gerenciamento de sessão, ao mecanismo de controle de concorrência (permitindo a escolha da granularidade da trava e a sua liberação) e ativação das ferramentas de suporte.

Por ser desenvolvido como uma aplicação Java, possui a característica da heterogeneidade de plataformas para execução, bastando que o ambiente de hardware/software disponível possua uma JVM (Java Virtual Machine) em funcionamento.

5. Conclusões

Este artigo apresentou aspectos e requerimentos para o projeto de um ambiente de groupware para escrita colaborativa e a sua integração com uma plataforma de suporte como *middleware* para distribuição. Esta plataforma, nomeada JAMP, é composta por um conjunto de frameworks que oferecem recursos comuns para reutilização em mais de uma aplicação com demanda por recursos semelhantes. Um destes frameworks, um broker, permite a comunicação transparente entre aplicações cliente/servidor desenvolvidos em linguagem Java. Em sistemas de co-autoria, requerimentos de design ainda são temas de pesquisa, principalmente no que diz respeito a generalização dos requisitos para sistemas de groupware síncronos, aplicações que apresentam dificuldades inerentes ao desenvolvimento de sistemas distribuídos.

A partir do levantamento destes requisitos, foi proposto um ambiente para escrita colaborativa denominado CoopWrite. Nele, são projetadas e implementadas as especificações para escrita conjunta consistente através de mecanismos de concorrência para sistemas de

groupware síncronos, e de ferramentas que promovem o conceito de consciência do trabalho em grupo e de comunicação inter-usuários fora do espaço de trabalho compartilhado.

Extensões para este trabalho consistem na implementação de canais de áudio e vídeo para a comunicação inter-usuários utilizando os frameworks de mídias contínuas da plataforma JAMP.

6. Referências Bibliográficas

[1] Baecker, R.M. & Nastos, D. & Posner, I.R. & Mawby, K.L. (1993) "The user-centred interative design of collaborative writing software." Em *Proceedings of the ACM INTERCHI Conference on Human Factors in Computing Systems,* pp 399-405, Amsterdam, 24-25 Abril 1993

[2] Baecker, R. & Glass G. & Mitchell, A & Posner, I. "SASSE: The Collaborative Editor", Em *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, Volume 2, pp. 459-470, 1994

[3] Dourish, P. & Bellotti, V. "Awareness and Coordination in Shared Workspaces". Em *CSCW'92: Proceedings of the Conference on Computer-Supported Cooperative Work.*Toronto: ACM Press, pp. 107-114.

[4] Ellis, C. & Gibbs, S. & Rein, G. "Groupware : Some Issues and Experiences", Em *Communications of the ACM*, vol 34, n.1, p.39-58, January 1991

[5] Ellis, C. & Gibbs, S. & Rein, G.; "Design and Use of a Group Editor." Em *Proceedings of the IFIP Engineering for Human-Computer Interaction Conference*. Pp. 13-25. 1989

[6] Ellis, C. & Gibbs, S.J. "Concurrency Control in Groupware Systems". Em *Proceedings of ACM SIGMOD 1989*. New York: ACM Press, pp. 343-355.

[7] Ferreira, M. M. & Trevelin, L.C. "A Plataform for Developing distributed Multimedia Applications", Relatório de Projeto - Departamento de Computação, UFSCar, 1998

[8] Fritzke, U. & Farines, J., "Um Suporte Para Aplicações de Trabalho Cooperativo Do Tipo Edtor Distribuído", Laboratório de Controle e Microinformática, UFSC

[9] Greenberg, S. & Marwood, D. "Real Time Groupware as a Distributed System: Concurrency Control and Its Effect on the Interface". *Em CSCW'94: Proceedings of the Conference on Computer-Supported Cooperative Work*. New York: ACM Press, pp. 207-217, Outubro 1994

[10] Greenberg, S. & Gutwin, C. " Sharing Fisheye Views in Relaxed-WYSIWIS Groupware Applications" Research Report 95/577/29, Department of Computer Science, University of Calgary, Canada, 1995

[11] Java Soft, Sun Microsystems. "Using Swing´s Text Components" *http://www.javasoft.com/docs/books/tutorial/uiswing/components/text.html*

[12] McGuffin, L. & Olson, G.M.. "ShrEdit, a Shared Electronic Workspace." Cognitive Science and Machine Intelligence Laboratory, University of Michigan. 1992

[13] Mitchell, A. "Communication and Shared Understanding in Collaborative Writing" M. Sc. Thesis at Toronto University, Canada. 1996

[14] Neuwirth, C.M. & Chandhok, R. & Kaufer, D.S. & Erion, P. & Morris, J. & Miller, D. “Flexible DIFF-ing in a Collaborative Writing System.” Em *CSCW `92: Proceedings of the Conference on Computer-Supported Cooperative Work*. Toronto: ACM Press, pp. 147-154.

[15] Olson, J.S. & Olson, G.M. & Mack, L.A. & Wellner, P. “Concurrent Editing: The Group’s Interface”. Em *Proceedings of Interact `90*. Pp. 835-840. 1990

[16] Rada, R. “Groupware and Authoring” Academic Press Limited, 1996

[17] Santos, A. C. & Ribeiro, S. V. & Meira, F. L. & Nakamura, A. & Alves, A. C. & Fagundes, L.G. "Adapting SACE-CSCW for Concurrent Engineering in the Internet", Em *Proceedings of CRIWG'99, Puerto Varas, Chile* - pp. 77-91 - Setembro 1996

[18] Vessoni, M. & Trevelin. L.C. "Aspectos de Design de Ferramentas para Escrita Colaborativa", Em *XXIV Conferencia Latinoamericana de Informatica CLEI'98 – Quito - Equador*, Vol. 2, pp 975-984, Outubro 1998