Usando Espaço de Tuplas para Criar um Editor de Escrita Colaborativa Móvel e Multi-Síncrono

Vaninha Vieira, Berthonio Lucena, Flávio Rocha

[1]Centro de Informática, Universidade Federal de Pernambuco
C.P. 7851, Recife, PE, Brasil – CEP 50732-970
[vvs,bml,fgr]@cin.ufpe.br

Abstract. Writing together is never easy. Collaborative writing tools can aid in the writing process by allowing, among other things, a shared common area used by many writers to do their work. However, the development of these tools can be difficult due to the need to deal with the environment complexities. Middlewares based on tuple spaces as a shared common area can fast this development. In this way, it was designed the CoWS, a collaborative writing editor that exploits the resources offered by a middleware based on tuple space that support a mobile environment, the Lime.

Resumo: Escrever em equipe nunca é fácil. Ferramentas de edição colaborativa podem auxiliar bastante o trabalho de escrita por oferecer, entre outras coisas, uma área comum compartilhada pelos diversos autores para a elaboração dos seus trabalhos. No entanto, o desenvolvimento dessas ferramentas pode ser difícil em virtude de precisar lidar com as complexidades de comunicação inerentes ao meio. Middlewares baseados em espaços de tuplas como área comum de compartilhamento podem facilitar esse desenvolvimento. Com base nisto, foi desenvolvido o CoWS, editor colaborativo que explora os recursos oferecidos por um middleware baseado em espaço de tuplas voltadas para ambientes móveis, o Lime.

1. Introdução

Os avanços crescentes nas tecnologias de comunicação vêm proporcionando ferramentas computacionais [1] que modificam significativamente o modo como as pessoas trabalham em equipe ou colaborativamente. Apesar da colaboração via computador poder ser obtida por meio de chat, e-mail, videoconferência etc., o desenvolvimento de ferramentas voltadas ao trabalho cooperativo/colaborativo suportado por computador (*Computer Supported Cooperative Work* - CSCW) tem sido alvo de intensas pesquisas em virtude das facilidades que introduzem em um ambiente colaborativo.

O processo mais estudado de autoria colaborativa é o de produção de textos [2]. A utilização da tecnologia computacional na escrita colaborativa deu origem a uma área de pesquisa chamada de Escrita Colaborativa Mediada por Computador (*Computer*

Supported Collaborative Writing) [3]. O presente trabalho leva em consideração aspectos da produção de textos em colaboração, apesar de o processo de autoria colaborativa não se restringir apenas a textos e poder ser estendido para outras formas de realização intelectual como as gráficas, as musicais e inclusive computacionais como especificações de sistema e programas [4].

A edição colaborativa pode ser definida como um processo no qual autores (editores, usuários ou revisores) com diferentes habilidades e responsabilidades interagem durante a elaboração ou a revisão de um documento. Para que a escrita colaborativa seja possível é necessário que exista um espaço compartilhado onde o texto possa ser acessado pelas pessoas que o estão desenvolvendo. A tecnologia de espaço de tuplas pode ser utilizada para prover essa área comum. Com base nisto, foi desenvolvido um editor colaborativo de texto, chamado CoWS (**Co**llaborative **W**riting through shared **S**paces), que utiliza o espaço de tuplas Lime [5] como meio de armazenamento.

Dentro desse contexto, o presente artigo visa apresentar a implementação de uma ferramenta computacional de edição colaborativa de texto que utiliza um espaço de tuplas para implementar o meio compartilhado visando, com isso, demonstrar que um modelo de coordenação baseado no espaço de tuplas é ideal para representar, interagir e compartilhar partes de um texto que está sendo editado de forma colaborativa.

Este artigo está organizado da seguinte forma: a próxima seção apresenta uma breve introdução sobre as principais características das ferramentas colaborativas. A seção 3 descreve a tecnologia de espaço de tuplas abordando o Linda e o Lime. A seção 4 procura relacionar as características do CoWS com as facilidades oferecidas pelo Lime. Em seguida, na seção 5 é descrito o funcionamento geral do protótipo desenvolvido para validar os conceitos levantados na especificação do CoWS. Por fim, a seção 6 conclui este artigo com as considerações finais.

2. Ferramentas Colaborativas

Escrever em equipe ou colaborativamente nunca é fácil [2]. A produção colaborativa de textos é o processo mais estudado de autoria colaborativa [4]. Muita pesquisa tem sido feita para investigar o modo como as pessoas escrevem colaborativamente, como elas aprendem a fazê-lo, e como as ferramentas baseadas em computador podem ajudar com os processos de escrita e de aprendizagem [6]. Assim, muitas teorias de escrita colaborativa foram criadas para caracterizar o processo de escrita como também diversas ferramentas foram desenvolvidas [2].

Apesar de já existirem muitas ferramentas que suportam escrita colaborativa para diferentes tipos de mídia, por exemplo, gráficos e documentos estruturados [1], pesquisas mostram que elas ainda não são muito usadas [7]. O principal objetivo dessas ferramentas é o de incentivar os participantes a trabalharem em grupo num documento comum, respeitando as características individuais [1].

As ferramentas de edição colaborativa devem possuir funcionalidades e características que facilitem o desenvolvimento de textos entre autores que estejam interagindo de forma colaborativa. O controle de concorrência, a suporte à percepção e o tipo de arquitetura da ferramenta (distribuída ou centralizada) são importantes características que devem ser levadas em consideração no projeto de tais ferramentas.

Além disso, o modo de interação entre os usuários, síncrono (no mesmo instante de tempo) ou assíncrono (em instantes diferentes) também é uma questão relevante.

O controle de concorrência visa garantir que não ocorrerá conflitos entre os usuários uma vez que os mesmos realizam operações simultâneas em documentos compartilhados [1]. Já a percepção possibilita que alterações feitas no texto por um autor sejam propagadas para todos os outros autores. Nas ferramentas que possibilitam que os usuários possam trabalhar assincronamente, desconectados ou em modo offline, nos seus textos, a percepção não é imediata pois os seus trabalhos só serão visualizados no momento em que houver a conexão e o respectivo compartilhamento.

Por fim, no projeto de uma ferramenta colaborativa pode-se optar por uma arquitetura distribuída ou centralizada. Na primeira, o armazenamento dos documentos ocorrerá em diversos hosts. Tipicamente o servidor armazena o documento e os clientes possuem uma cópia desse documento ficando a cargo do controlador de versão verificar as atualizações existentes. Com isso, os clientes podem trabalhar independentemente do funcionamento da rede. Agora, na arquitetura centralizada os documentos são armazenados em um servidor central que fica responsável por manter os arquivos compartilhados consistentes. Uma desvantagem dessa abordagem é que se o servidor não funcionar, os clientes não poderão trabalhar mais.

3. Implementação de Espaço de Tuplas para Ambientes Móveis

A computação móvel define um cenário desafiante e muito dinâmico para o desenvolvimento de software [8, 9]. O desenvolvimento de aplicações que apresentam mobilidade física e/ou lógica é bastante complexo em virtude das constantes mudanças na disponibilidade de recursos do ambiente (dados e elementos computacionais) [1]. Uma alternativa para simplificar a programação é o uso de um middleware que lide com as complexidades e mudanças nesses ambientes deixando que o programador concentre seus esforços na aplicação ao invés de pensar em detalhes do ambiente que mudam constantemente.

Middleware baseados em modelos computacionais que empregam memória compartilhada, como o Linda, vem sendo usados como uma alternativa viável para simplificar o desenvolvimento de aplicações distribuídas. Apesar do modelo do Linda apresentar várias implementações (JavaSpaces [10], Tspaces [11] etc.), ele não foi projetado para suportar ambientes móveis [1]. Uma adaptação desse modelo para ambientes móveis resultou no Lime.

3.1 Linda

O modelo computacional do Linda foi proposto no início da década de oitenta como um novo modelo de comunicação entre processos [12]. Nesse modelo, os processos se comunicam através da escrita, leitura e remoção de dados em uma área de memória compartilhada chamada espaço de tuplas. Um espaço de tuplas é um repositório global e persistente de tuplas que são estruturas de dados essenciais

constituídas por uma seqüência ordenada de campos. As tuplas contêm as informações reais que estão sendo comunicadas. Cada processo concorrente no sistema tem acesso ao espaço de tuplas (propriedade global) que existe independentemente da existência dos processos (propriedade de persistência).

O modelo do Linda fornece coordenação entre os componentes por meio do compartilhamento do seu espaço de tuplas. Nesse modelo há o desacoplamento espacial e temporal. O desacoplamento espacial estabelece que os componentes não precisam coexistir ao mesmo tempo no sistema para se comunicar. Por sua vez, o desacoplamento temporal especifica que um dado componente pode residir em qualquer lugar do sistema distribuído.

Como as tuplas são anônimas, a sua pesquisa se dá através da observação de seus conteúdos por meio de um *template*. Um *template* é uma tupla cujos campos contêm valores reais ou formais. Valores reais especificam qualquer valor que pode ser armazenado em um dado campo ao passo que os valores formais atuam como curingas representando, assim, uma faixa de valores possíveis para um dado campo de acordo com o seu tipo. Na seleção de uma tupla os valores do *template* são comparados aos valores das tuplas armazenadas até que uma tupla com valores idênticos seja encontrada.

A manipulação das tuplas no espaço é feita com apenas três operações: escrever uma tupla no espaço (*out*), ler uma tupla do espaço de acordo com um *template* (*rd*) e retirar uma tupla do espaço de acordo com um *template* (*in*). No caso de existir várias tuplas correspondentes a um dado template, apenas uma é retornada de forma não determinística. As últimas duas operações (ler e retirar) são bloqueantes, isto é, se uma tupla não for encontrada, o processo que solicitou é suspenso até que uma tupla que satisfaça a pesquisa apareça no espaço. No caso de múltiplos processos bloqueados para retirar uma tupla, quando a tupla aparece, apenas um processo selecionado de forma não determinística receberá a tupla. Os demais receberão uma cópia da tupla.

As características do Linda são adequadas para configurações móveis [9]. A comunicação é implícita (através do espaço de tuplas) e há desacoplamento espacial e temporal. Este desacoplamento é de fundamental importância em ambientes móveis, onde as partes comunicantes movimentam-se dinamicamente em virtude de migrações. Uma evolução natural do modelo de coordenação do Linda seria o suporte a ambientes móveis.

3.2 Lime

Lime (*Linda in a Mobile Environment*) é um middleware projetado para possibilitar o desenvolvimento rápido de aplicações que apresentam mobilidade física (*hosts*) e/ou lógica (agentes) [8]. Ele adapta o espaço de tuplas do Linda [13] para ambientes móveis através da quebra da noção de um espaço de tuplas global, ou seja, o conteúdo encontra-se distribuído através dos múltiplos componentes móveis.

A coordenação das unidades móveis lógicas e físicas ocorre pela exploração de um espaço de tuplas transiente e reativo cujos conteúdos mudam de acordo com a conectividade [8]. Para isso, o modelo de coordenação e comunicação do Linda passou a englobar tanto a mobilidade física (*hosts*) quanto a lógica (agentes). Neste modelo, agentes são os únicos componentes ativos do sistema e os únicos que possuem um espaço de tuplas real. Já os *hosts* são apenas repositórios para os agentes

ficando responsáveis por fornecer-lhes suporte à conexão e execução. Os agentes podem mover-se através de *hosts* móveis que, por sua vez, podem mover-se através do espaço físico.

As propriedades do ambiente de um componente móvel forma o seu contexto. Quando a mobilidade é totalmente explorada não há contexto fixo, global e predefinido para a computação como assumido por Linda [8]. Em vez disso há um contexto global e dinâmico definido pelos diferentes contextos individuais das unidades móveis que estão conectadas. O modo como a unidade móvel interage nesse contexto dinâmico não muda porque operações básicas ocorrem usando as mesmas primitivas, como em Linda.

3.2.1 Espaço de Tuplas Federado

A presença de mobilidade impede a existência de um espaço de tuplas persistente e global, entretanto, cada agente móvel é dono de pelo menos um Espaço de Tuplas Lime (*Lime Tuple Space* – LTS) que segue o agente durante a migração [1]. O compartilhamento do LTS é uma poderosa abstração que fornece à unidade móvel a ilusão de um espaço de tuplas local contendo tuplas de todas as unidades atualmente conectadas sem precisar saber quais são [9]. Em outras palavras, uma operação *in* requisitada por qualquer um dos agentes pode retornar uma tupla correspondente do espaço de tuplas do Lime de qualquer agente. Este espaço de tuplas tem o nome de espaço de tuplas federado (Fig. 1).

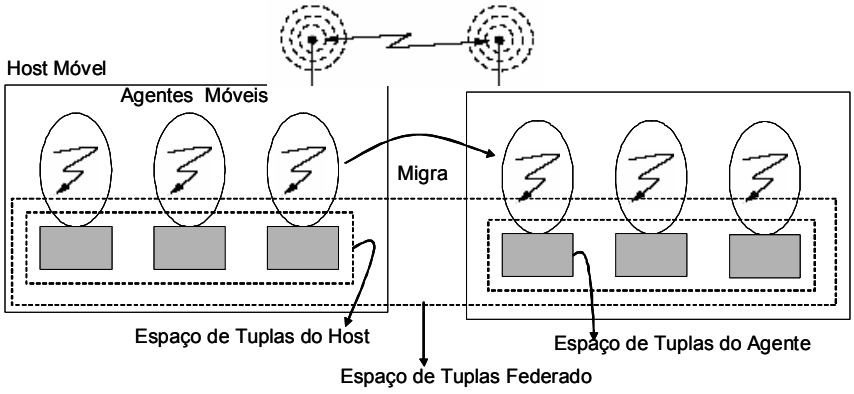

Fig. 1. Espaço de Tuplas Federado do Lime

3.2.2 Localização de Tuplas

O espaço de tuplas do Lime é um sistema naturalmente distribuído onde as tuplas são fisicamente armazenadas nos múltiplos *hosts* [5]. De fato, cada tupla está associada a um único agente e armazenada no mesmo *host* onde este agente está residente.

Caso o programador precise de um controle maior de onde as tuplas são armazenadas e de onde as operações de busca são realizadas, o Lime estende as

operações de Linda com parâmetros de escopo expresso em termos de identificadores de agentes ou hosts que restringem as operações a uma parte do espaço de tuplas compartilhado [9]. A operação `out(t)` foi estendida na operação `out(1,t)` através do acréscimo de um parâmetro de localização representando o identificador do agente responsável em armazenar a tupla. Foram também criadas variantes das operações `in(p)` e `rd(p)`, as operações `in(w,1,p)` e `rd(w,1,p)`, respectivamente, onde w indica a tupla que realiza a operação, o 1 corresponde ao identificador do agente cujo espaço será pesquisado em busca da tupla que corresponde ao *template* p passado como parâmetro.

3.2.3 Engajamento/Desengajamento de Hosts

Em um ambiente móvel, mudanças na conectividade entre os *hosts* são ocorrências normais à medida que os componentes se movem para dentro ou para fora da faixa de alcance uns dos outros [5]. Todos os *hosts* e agentes no alcance da conectividade podem compartilhar seus espaços de tuplas. O processo de um *host* vir ao alcance é chamado de engajamento (o *host* deseja compartilhar seu espaço com outros *hosts* na rede) enquanto a saída do alcance é o desengajamento. Tanto o engajamento quanto o desengajamento são operações atômicas. Em outras palavras, todos os *hosts* vêem a mudança na configuração do sistema na mesma ordem. O engajamento ou desengajamento pode ser realizado de duas formas: sinalização ou distância segura.

a) Engajamento/Desengajamento via Sinalização

O método da sinalização é baseado na noção de detecção de um sinal periódico de um *host* que ainda não é parte de um grupo. Esta detecção aciona o processo de engajamento. Se um *host* deseja desconectar-se de um grupo ele pára de sinalizar antes de sair da faixa de alcance do grupo. Isto permite que o processo de desconexão ocorra antes do *host* tornar-se inacessível. Este mecanismo é útil em configurações *ad hoc* tais como pequenos grupos de encontros.

b) Engajamento/Desengajamento via Distância Segura

O segundo tipo de mecanismo de engajamento fornecido pelo Lime permite o engajamento de *hosts* (ou grupos de *hosts*) baseados no critério que define uma distância segura. Ela é definida baseada no alcance de comunicação dos dispositivos, suas velocidades máximas, e um número de outros fatores físicos. Em um alto nível, a distância segura é essencialmente definida como a distância entre os *hosts* (informação de GPS) na qual a comunicação pode ser completada antes que os *hosts* não possam mais se comunicar. Isto assegura que qualquer comunicação que o Lime começa poderá ser completada antes que os participantes se desconectem.

3.2.4 Reação

Mobilidade possibilita um ambiente altamente dinâmico onde reações às mudanças constituem uma fração maior do projeto da aplicação [5]. O Lime estende o espaço de tuplas do Linda com a noção de reação. Uma reação é definida como um fragmento de código que especifica ações que devem ser executadas quando uma pesquisa de tupla é satisfeita [9]. *Hosts* podem apenas reagir às tuplas nos espaços de tuplas que estão engajados e compartilhados.

As reações podem ser fortes ou fracas. Reações fortes agrupam em uma operação atômica os passos de detecção da tupla e a execução do código correspondente (modo síncrono). Ela necessita da declaração de uma localização física onde o evento deve ocorrer e, por isso, fica limitada pela mobilidade física dos *hosts*. Por sua vez, as reações fracas separam os passos de detecção e de execução permitindo que a execução ocorra em algum tempo após a detecção (modo assíncrono). Na reação fraca não há necessidade de especificar onde o evento ocorrerá. Assim, o programador pode descrever, uma única vez, todas as ações que serão executadas em resposta a um evento, sem se preocupar com mudanças que possam vir a acontecer na configuração do sistema [8].

O processamento de uma reação fraca é similar ao processamento de uma reação forte exceto que a execução do método não ocorre sincronamente quando uma tupla é encontrada. Reações fortes são úteis para serem executadas localmente no *host* ao passo que as reações fracas são adequadas para serem usadas nos *hosts* remotos e, com isso, no espaço de tuplas federado [9].

4. O Editor Colaborativo CoWS

O CoWS (***Co****llaborative* ***W****riting through shared* ***S****paces*) é um editor de texto colaborativo multi-síncrono voltado para ambientes móveis. A sua funcionalidade multi-síncrona permite que ocorram interações síncronas e/ou assíncronas dos autores na elaboração do texto. CoWS utiliza o Lime para explorar todos os recursos que a camada de coordenação desse *middleware* oferece para desenvolver aplicações que exibem mobilidade física e/ou lógica. Essa camada de coordenação do Lime é suportada por um modelo de memória compartilhada conhecido por espaços de tuplas.

4.1 Editor Colaborativo e Lime

Nesta subseção são apresentadas algumas questões que foram relevantes no projeto do CoWS para adequá-lo ao Lime e discute como a abordagem de espaço de tuplas contribuiu para facilitar a implementação desse tipo de aplicação.

a) Papéis na Escrita Colaborativa

Um grupo formado para realizar escrita colaborativa pode ser composto por diversos papéis, como autor, revisor, coordenador, co-autor, editor, etc. A versão atual do CoWS apóia as atividades de múltiplos autores. Os autores são aqueles que contribuem com a escrita do texto, elaborando ou modificando partes do texto. Em trabalhos futuros o editor pode ser estendido para apoiar outros papéis e atividades seguindo um processo de fluxo de trabalho.

b) Parágrafos e Tuplas

No editor, cada tupla armazenada no espaço de tuplas contém um parágrafo de um texto em edição. O conteúdo de um parágrafo pode ser uma seqüência de caracteres

alfa-numéricos e símbolos especiais do padrão ANSI ou uma imagem. O protótipo implementado, entretanto, não considera o tratamento de parágrafo com imagem, o que pode ser abordado em trabalhos futuros.

O documento compartilhado pelos autores no CoWS é o texto, composto por vários parágrafos que são as unidades atômicas de adição, edição e remoção. Essa coleção de parágrafos segue uma determinada ordem. Armazenar cada parágrafo apenas em uma tupla não é suficiente para recuperar o texto ordenado. Portanto, algumas informações relacionadas à ordem dos parágrafos devem ser armazenadas nas tuplas. Essas e outras informações adicionais sobre as tuplas serão detalhadas adiante.

c) Modo Multi-síncrono e Modelo de Coordenação

No CoWS, os autores podem trabalhar engajado (modo síncrono) ou desengajado (modo assíncrono). No modo síncrono, as alterações dos autores serão percebidas tão logo ocorram. No modo assíncrono, as alterações realizadas por um autor só serão visíveis aos demais autores que estiverem conectados quando ele se engajar. Garantir o trabalho assíncrono é fundamental em cenários móveis onde a desconexão dos *hosts* é uma constante.

Os autores podem se conectar ao sistema, olhar o estado atual do texto, selecionar os parágrafos que desejam remover ou alterar e então se desconectar. Em modo assíncrono podem fazer suas alterações no texto, através da inclusão de novos parágrafos, remoção ou alteração dos existentes, que tenham sido previamente selecionados. Dessa maneira, o texto pode ser construído de forma independente e paralela e os resultados intermediários podem ser novamente compartilhados quando a conectividade é restabelecida.

Esta interação distribuída síncrona e assíncrona é obtida naturalmente por causa do modelo de coordenação dos espaços de tuplas: o total desacoplamento espacial e temporal.

d) Perspectiva Fracamente Consistente e Mobilidade

O editor compartilhado exibe todos os parágrafos que estavam no espaço quando da sua última conexão mesmo se estiverem selecionados por outros autores. Com isso, a perspectiva exibida no espaço de trabalho de um autor em modo assíncrono é apenas fracamente consistente com o estado global do texto, uma vez que ela representa a última informação conhecida sobre o texto. Uma consistência total é difícil de ser garantida considerando as freqüentes desconexões presentes em um cenário móvel.

e) Percepção e Reação

A percepção de mudanças no texto é implementada com reações. O CoWS reage às tuplas que são inseridas no espaço de tuplas federado. Logo, em modo assíncrono, o agente só reage ao espaço local. O tipo de reação que é fornecida pelo Lime e é utilizada pelo CoWS é a reação fraca. Este tipo de reação reage apenas à chegada de uma nova tupla no espaço, independente de mudanças que possam vir a acontecer na disposição do sistema.

f) Controle de concorrência e Localização de tuplas e Desengajamento

As políticas de controle de acesso definem uma área exclusiva para interação do texto sem a interferência de outro usuário, que pode ser ao nível de seções, parágrafos ou letras, sendo que quanto menor o nível de granularidade definido, maior controle será necessário, demandando maior largura de banda [1].

O nível de granularidade do CoWS são os parágrafos e o tipo de controle de concorrência implementado é o bloqueio. O mecanismo de bloqueio, implementado pelo editor, funciona da maneira a seguir. O Lime possui suas operações estendidas com parâmetros de localização das tuplas, permitindo acesso a partes do espaço de tuplas compartilhado transientemente. Quando um parágrafo é criado, ele é inserido no espaço de tuplas local do agente que o criou. Esse agente passa a ser o dono do parágrafo e detém o controle do mesmo. Quando é feita uma conexão ao espaço de tuplas federado, os espaços de tuplas dos demais agentes passam a enxergar esse parágrafo sem, entretanto, poder modificá-lo. Caso um outro usuário deseje modificar ou remover o parágrafo bloqueado, ele deverá primeiro selecionar o parágrafo. Com isso, a tupla correspondente ao parágrafo é removida do espaço de tuplas onde está localizado fisicamente e é transferida para o espaço de tuplas local do agente que solicitou a seleção. Dessa maneira, o parágrafo passa a residir fisicamente nesse novo espaço de tuplas e operações realizadas por outros agentes não terão efeito sobre esse parágrafo. Porém, o bloqueado sobre ele é apenas parcial, porque outros usuários podem selecioná-lo a qualquer momento. Para garantir o bloqueio total de seus parágrafos selecionados, os usuários devem desconectar do editor.

g) Representação do Texto e Tuplas do espaço

Uma das questões principais na utilização da abordagem de espaço de tuplas na modelagem do editor compartilhado CoWS é como montar o texto final a partir das diversas tuplas espalhadas pelo espaço, garantindo coerência com as intenções dos diversos autores. O espaço de tuplas não guarda nenhuma ordem das tuplas inseridas nem dos parágrafos, o que poderia levar a uma inconsistência na remontagem do texto entre cada um dos usuários. Pensando em diversos autores trabalhando de forma síncrona e assíncrona, modificando e produzindo diferentes versões do texto, criou-se a idéia de uma estrutura de dados que foi denominada lista não encadeada. Lista porque possui anterior e próximo. Não encadeada porque o anterior e próximo não são necessariamente os vizinhos imediatos. Com isso, o domínio do problema de inconsistência, que era distribuído, passou a ser apenas o de manter consistente uma lista local. O algoritmo que foi utilizado para manter essa lista local consistente será discutido na subseção 5.2.

Logo, o modelo das tuplas que representam os parágrafos contém as seguintes informações:

- **Id**: Representa o identificador único de parágrafo possibilitando sua posterior recuperação do espaço de tuplas. Esse id é formado pelo *login* do usuário que o criou seguido de um seqüencial;
- **Head**: Indica o id do parágrafo de onde o parágrafo corrente foi criado. Null, se o parágrafo for inserido como o primeiro do texto;
- **Tail**: Armazena o id do parágrafo de referência anterior ou posterior ao parágrafo origem; Null, se o anterior ou posterior não existir;
- **Direction**: indica a direção de inserção do parágrafo dentro do texto. Assume os valores *true/false*. Se *true*, o Tail indica o parágrafo anterior. Se false, Tail indica o posterior;
- **Text**: O conteúdo do parágrafo;
- **Selection**: Indica o *login* do usuário que detém o controle do parágrafo;
- **Delected**: *Flag* que indica se o parágrafo está ativo ou inativo no texto. Um parágrafo removido não é retirado imediatamente do espaço de tuplas. Ele fica marcado como inativo, permitindo que uma posterior recuperação seja realizada. Uma rotina de retirada de lixo pode de tempos em tempos, remover os parágrafos marcados como inativos.

Na versão atual do CoWS é considerada apenas a elaboração de um único texto em um espaço de tuplas. Porém, essa abordagem pode ser estendida permitindo o gerenciamento de múltiplos textos no mesmo espaço de tuplas, usando para isso um nome único que identifique o texto na tupla de parágrafo. Além disso, podem-se acrescentar outras informações como anotações e revisões do texto, através da criação de novas tuplas, atreladas às tuplas parágrafo.

h) Arquitetura do CoWS e Arquitetura do Lime

O CoWS possui uma arquitetura totalmente distribuída, com o texto global em produção sendo armazenado em diversos *hosts* e agentes, como mostra a Fig. 2. Cada agente detém seu espaço de tuplas local, onde suas tuplas são armazenadas. O conjunto dos espaços de tuplas locais engajados compõe o espaço de tuplas federado. Existe mobilidade física dos hosts mas não há mobilidade lógica dos agentes.

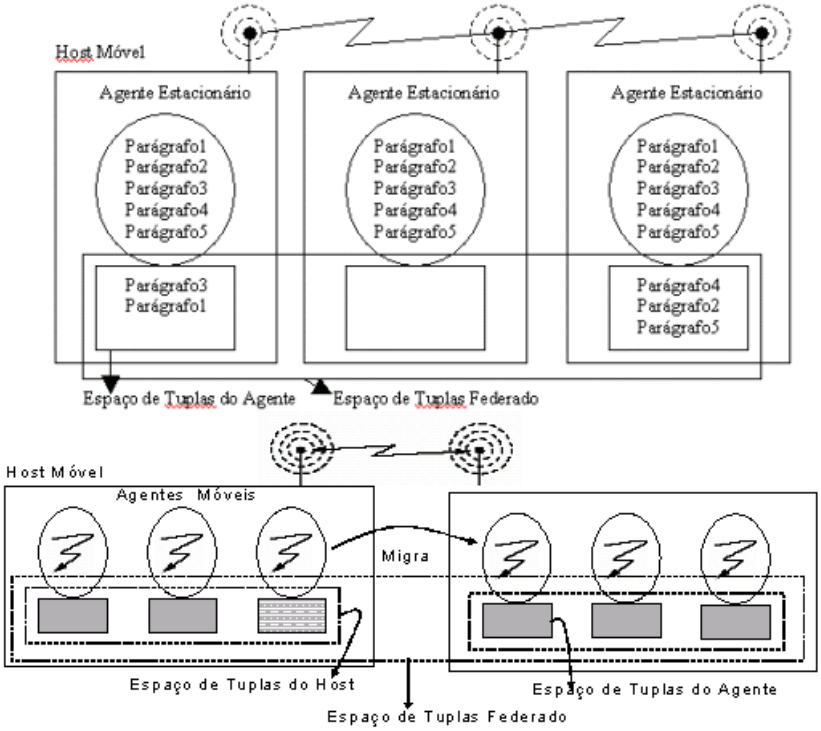

Fig. 2. Arquitetura geral de funcionamento do CoWS, figura de cima, seguindo o modelo do Lime, figura de baixo

5. Protótipo do CoWS

Para validar as idéias por trás do uso do Lime no desenvolvimento do editor colaborativo CoWS, implementamos um protótipo do editor. Essa versão do protótipo tem por objetivo resolver a comunicação e uso do espaço de tuplas como espaço de trabalho compartilhado, considerando cenários de colaboração multi-síncronos (onde os usuários podem trabalhar tanto de forma síncrona quanto assíncrona), mantendo a consistência do texto nas áreas de trabalho dos diversos autores. Dessa maneira, aspectos específicos de interface e formatação de texto não foram o foco principal e serão explorados em versões posteriores do editor.

A implementação foi feita usando a linguagem Java, para garantir compatibilidade com o Lime, e também devido ao suporte dessa linguagem para execução em várias plataformas e disponibilidade de recursos avançados de computação distribuída. A comunicação é gerenciada pelo Lime, utilizando sockets. A implementação de espaço de tuplas utilizada é o LighTS [14], uma implementação leve de espaço de tuplas, disponibilizada com a versão de distribuição do Lime. Os testes foram executados com os hosts rodando sobre Windows2000.

A seção seguinte apresenta a interface do editor e detalha seu funcionamento geral, apresentando um cenário de trabalho multi-síncrono como exemplo de sua utilização. E a próxima seção apresenta o algoritmo de ordenação de tuplas criado para gerenciar a ordem em que os parágrafos devem aparecer no texto.

5.1 Interface e Funcionamento Geral

O CoWS provê acesso aos diversos parágrafos que compõe o texto aos vários autores. Cada autor pode contribuir com a edição do texto inserindo novos parágrafos, em diferentes posições (antes ou após um determinado parágrafo, no início ou no final do texto), alterando e removendo parágrafos existentes. Nessa versão não foi implementada a movimentação de parágrafos, ficando como trabalhos futuros.

Os autores podem selecionar um ou mais parágrafos nos quais deseja trabalhar. Os parágrafos criados pelo autor e/ou selecionados por ele ficam armazenados em seu espaço de tuplas local. Esses parágrafos aparecem em seu editor em fonte preta, enquanto os parágrafos pertencentes a outros usuários aparecem em cinza. Essa informação aumenta a percepção dos usuários, auxiliando-os a identificar quais parágrafos estão em seu poder e o que ele pode alterar e remover, quando estiver desconectado. Uma outra informação de percepção é o nome do usuário que criou o parágrafo, que aparece ao lado de cada parágrafo criado.

Uma ilustração da interface principal do CoWS é exibida na
Fig. **3**. A interface do CoWs está dividida em duas partes principais: o Editor Colaborativo e um Chat comum que permitem que autores enviem mensagens de texto aos demais autores conectados. O Editor, por sua vez, está dividido em três partes. Uma área de texto editável na parte inferior, onde os parágrafos são editados. Uma área de texto não editável na parte superior, que contém os parágrafos ordenados que irão compor o texto final. Nessa área, cada parágrafo é identificado pelo login do autor e por um número sequencial. Por fim, o editor apresenta na lateral do lado esquerdo, uma área que exibe uma lista com os identificadores dos parágrafos, que permite que o autor selecione o parágrafo para seu espaço de tuplas local, para alteração e/ou remoção.

O texto pode ser visualizado de duas maneiras: no formato apresentado na
Fig. **3**, com uma visão mais simplificada apenas com a identificação do parágrafo e seu conteúdo. E em um modo detalhado, exibindo todo o conteúdo da tupla, tal como ela está fisicamente armazenada no espaço de tuplas.

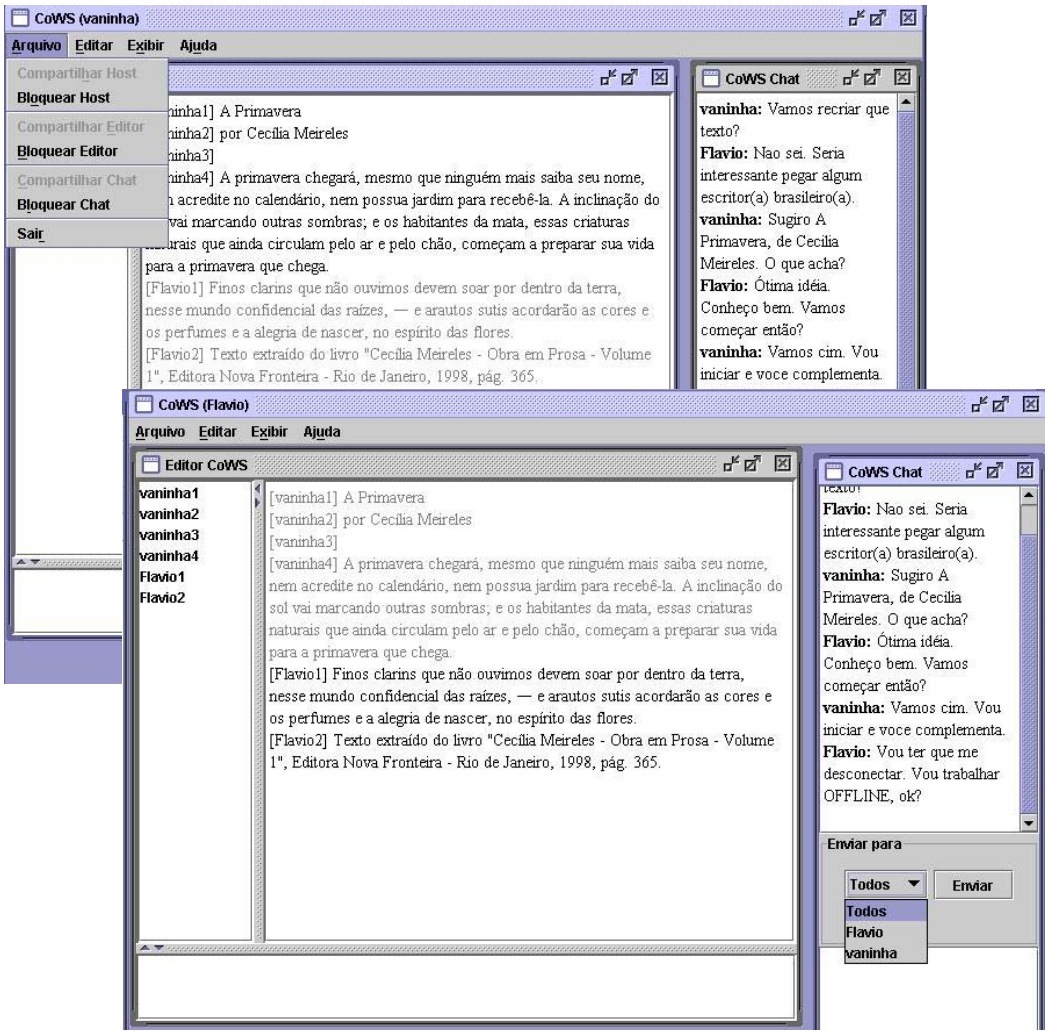

Fig. 3. Tela Principal do CoWS – Dois usuários interagindo em modo síncrono, através do Editor e do Chat

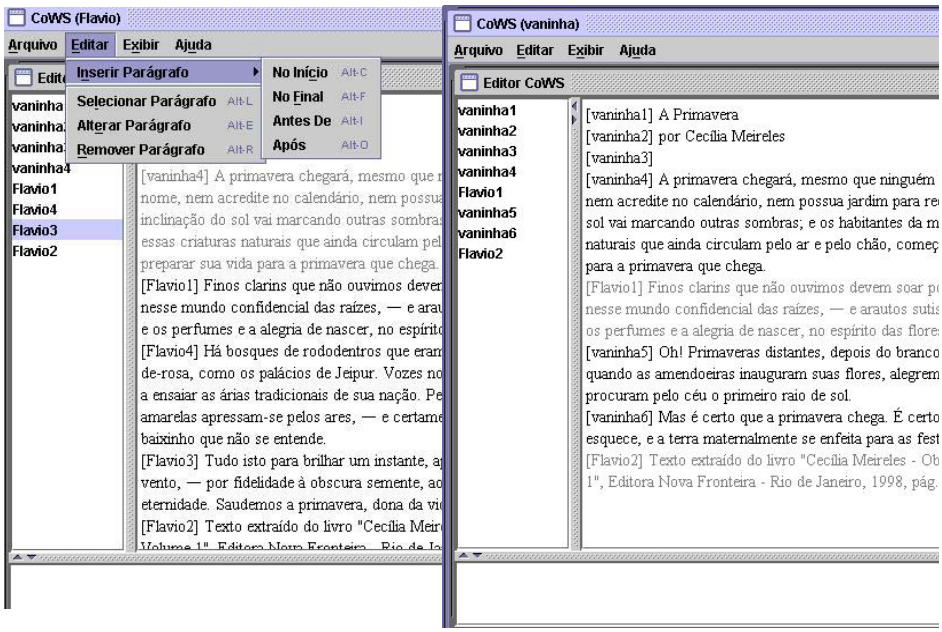

Fig. 4. Visão parcial do CoWS – Dois usuários trabalhando de forma assíncrona

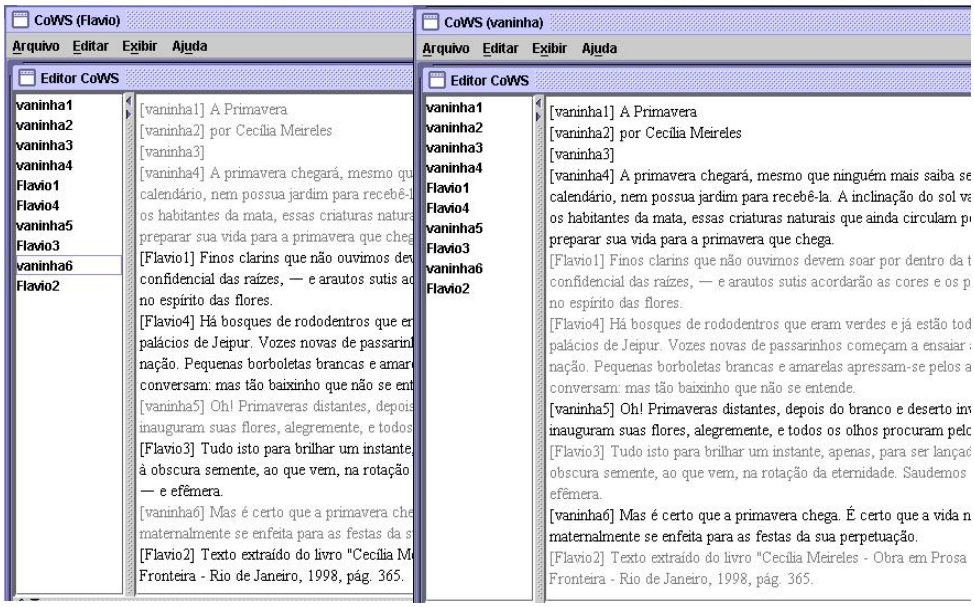

Fig. 5. Visão parcial do CoWS – Dois usuários retornando ao modo síncrono, compartilhando trabalhos realizados em modo assíncrono

Como um cenário de trabalho colaborativo e utilização do editor, mostramos na Fig. **3** dois usuários interagindo através do Chat para criar um novo texto. Nesse primeiro cenário, eles trabalham de forma síncrona, compartilhando o espaço. O compartilhamento do Editor e do Chat é feito através do menu Arquivo, como mostra a
Fig. **3**. Todos os parágrafos inseridos por um autor ficam imediatamente visíveis ao outro autor. O segundo cenário, ilustrado na Fig. 4, é a realização de trabalho em modo assíncrono, onde os dois usuários estão desconectados do espaço, e trabalham de forma isolada, sem que um enxergue as alterações no texto que estão sendo feitas pelo outro. E, por fim, o terceiro cenário, apresentado na Fig. 5, mostra os dois usuários retornando ao espaço compartilhado, onde os parágrafos que foram inseridos em modo assíncrono por cada um são incorporados ao texto final. Observe que os dois passam a ter a mesma visão do texto final. Essa visão uniforme é alcançada através de um algoritmo de ordenação de tuplas que será explicado na próxima seção.

5.2 Algoritmo de Ordenação de Tuplas

Uma dificuldade no projeto do CoWS foi garantir a ordem do texto sendo produzido, considerando as intenções de inserção dos autores que estejam trabalhando de forma síncrona ou assíncrona. Com a criação da estrutura de dados lista não encadeada, este problema de consistência passou de um contexto distribuído para um contexto local. Para que cada agente mantenha sua lista consistente uns com os outros, basta que cada um deles utilize um mesmo algoritmo de consistência de lista. A responsabilidade de cada agente agora seria em manter a sua lista consistente.

```
Varre a lista não encadeada procurando o head e o tail
Se direction for true
     Procurar o parágrafo entre head e tail, originado de head,
     cujo id é menor do que o id do novo parágrafo e se existir
     colocar em head
Senão
     Procurar o parágrafo entre head e tail, originado de head,
     cujo id é maior do que o id do novo parágrafo e se existir
     colocar em head
Procurar o parágrafo mais distante de head, entre head e tail, e
que se originou de head e colocar em newHead
Se newHead existir
    head é igual a newHead
    repete
Senão
    sair
Se direction for true
    Insere o novo parágrafo antes de head
Senão

    Insere o novo parágrafo depois de head
```

Fig. 6. Algoritmo de Ordenação das Tuplas

O algoritmo de consistência de lista recebe como entrada uma nova tupla a ser inserida na lista não encadeada, que na visão do usuário é a réplica do texto que cada um dos agentes possui. Se a réplica do texto não possui o anterior ou o posterior do novo parágrafo, ele não é inserido até que seu anterior e posterior cheguem. Se a réplica do texto possui anterior ou posterior, o algoritmo apresentado na Fig. 6 é executado.

A grande vantagem dessa abordagem é que não é necessário realizar bloqueio sobre nenhum parágrafo do texto para realizar a inserção de novos parágrafos.

6. Conclusões

A escrita colaborativa é a principal área de pesquisa da autoria colaborativa. Diversas teorias e ferramentas têm surgido para auxiliar e simplificar o trabalho de escrita colaborativa de um texto. Como as ferramentas de edição colaborativa devem tratar as complexidades inerentes ao meio de comunicação usado pelos autores, o seu desenvolvimento não é fácil. Além de ter que fornecer uma área compartilhada usada para armazenar o texto, essas ferramentas têm que implementar um modelo de coordenação que possibilite, entre outras coisas, políticas de acesso e de bloqueio a essas áreas.

O uso de *middlewares* voltados para ambientes distribuídos podem otimizar o desenvolvimento dessas ferramentas uma vez que os mesmos possibilitam que o programador foque nos detalhes da aplicação ao invés de se preocupar com a comunicação no meio. Esses *middlewares* podem ser construídos explorando os

benefícios oferecidos pelo espaço de tuplas, como o desacoplamento espacial e temporal.

A ferramenta de edição de texto colaborativa ora proposta, o CoWS (***Co****llaborative* ***W****riting through shared* ***S****paces*), foi concebida por meio da utilização de um *middleware* voltado para ambientes móveis, o Lime. O objetivo principal foi o de validar as idéias do Lime para o desenvolvimento de uma ferramenta de edição colaborativa. Um protótipo foi implementado visando testar as questões de comunicação e uso do espaço de tuplas considerando cenários de colaboração multi-síncronos e a consistência do texto nos espaços de trabalho dos diversos autores. Dessa maneira, aspectos de interface e de formatação de textos não são o foco dessa versão do CoWS e deverão ser exploradas em versões posteriores do editor.

Em trabalhos futuros o editor colaborativo pode ser estendido para apoiar outros papéis além do autor e possibilitar o gerenciamento de múltiplos textos no mesmo espaço de tuplas. Além disso, com a criação de novas tuplas outras informações como anotações e revisões do texto poderão ser atreladas às tuplas parágrafo.

Referências

[1] Gonçalves, P. R.; Padilha, T. P. P. "Implementação de uma Ferramenta de Edição de Texto Colaborativa". Anais do V Encontro de Estudantes de Informática do Tocantins, pp. 353-360, Palmas, TO. Em: http://www.ulbrato.br/ensino/43020/artigos/anais2003/anais/edicaodetextocolaborativa-encoinfo2003.pdf

[2] Posner, I.R.; e Baecker, R.M. "How People Write Together". Anais da 25ª Conferência Internacional sobre Ciências de Sistemas. Volume IV, pp. 127-138. Janeiro de1992.

[3] Lordello, M. "Tecnologias da Infra-estrutura de Informação em Ambientes Colaborativos de Ensino". Monografia Final. Em: http://www.dca.fee.unicamp.br/courses/IA368F/1s1998/Monografias/chaim.html

[4] Chang, K. H.; Murphy, L.; Fouss, J. D.; Dollar, II, T. D.; Lee, B. G. e Chang, Y. "Software Devolpment and Integration in a Computer Supported Cooperative Work Environment". Software-Practice and Experience, vol. 28, no. 6, pp. 657-679, Maio de 1998.

[5] Lime. Em: http://lime.sourceforge.net.

[6] Mitchell, Alex; Posner, Ilona; Baecker, Ronald. "Learning to Write Together Using Groupware". Em: http://kmdi.utoronto.ca/RMB/papers/p14.pdf.

[7] S., Noël; J.-M., Robert. "Empirical study on collaborative writing: What do co-authors do, use, and like?". Computer Supported Cooperative Work: The Journal of Collaborative Computing, 13 (1), 63-89.

[8] Murphy, A.L.; Picco, G.P; Roman, G.-C. "Developing Mobile Computing Applications with Lime". 22ª Conferência Internacional de Engenharia de Software (ICSE'00). Julho de 2000.

[9] Murphy, A.L.; Picco, G.P. "Using Coordination Middleware for Location-Aware Computing: A Lime Case Study". 6ª Conferência Internacional sobre Linguagens e Modelos de Coordenção (Coordination'04). Fevereiro de 2004.

[10] JavaSpaces. Em: http://www.jini.org/nonav/standards/davis/doc/specs/html/js-spec.html.

[11] IBM. TSpaces. Em: http://www.almaden.ibm.com/cs/TSpaces.

[12] Murphy, A.L.; Picco, G.P; Roman, G.-C. "LIME: Linda Meets Mobility". 21ª Conferência Internacional de Engenharia de Software. May 1999.

[13] Gelernter, D. "Generative Communication in Linda". ACM Computing Surveys. 7(1):80–112. Janeiro de 1985.

[14] LIGHTS. Em: lights.sourceforge.net.