Um *Framework* na Plataforma JAMP para suporte ao Trabalho Cooperativo.

Marcelo de Paiva Guimarães[1]
e-mail: paiva@dc.ufscar.br

Luis Carlos Trevelin
e-mail: trevelin@power.ufscar.br

Universidade Federal de São Carlos (UFSCar)
Department of Computing
Rod. Washington Luiz, Km 235
São Carlos (SP) – Brazil
13.5565-905 – Postal Box 676
Phone: +55 16 260 8233

Resumo

Este trabalho apresenta uma proposta da união de Trabalho Cooperativo, Aplicações Multimídia e Sistemas Distribuídos, dando origem a um *framework* para suporte ao Trabalho Cooperativo na plataforma JAMP (*Java Architecture for Media Processing*) e a um editor gráfico cooperativo para testar este *framework*. A plataforma JAMP é um ambiente de programação distribuída baseada em Java/RMI (*Remote Method Invocation*), muito adequada para o desenvolvimento de aplicações WEB. O *framework* de Trabalho Cooperativo é um projeto abstrato orientado a objetos que pode ser adaptado segundo as necessidades da aplicação.

Palavras-Chaves: Trabalho Cooperativo, *Frameworks*, Sistemas Distribuídos, Plataforma JAMP, WEB Design.

Abstract

This work presents a proposal for the union of Cooperative Work, Multimedia Applications and Distributed Systems, by originating a framework for supporting Cooperative Work in the JAMP (*Java* Architecture for Media Processing) platform and a cooperative graphic editor to test this framework. The JAMP platform is a distributed programming environment based on Java/RMI (Remote Method Invocation), very well adapted for WEB applications development. The Cooperative Work framework is an abstract object oriented project that can be adapted according to the applications needs.

Keywords: Cooperative Work, Frameworks, Distributed Systems, JAMP platform, WEB Design.

1. Introdução

A necessidade de um ambiente para ser utilizado na construção de aplicações que suportem o Trabalho Cooperativo e as Aplicações Multimídia, tem despertado o interesse na utilização de tecnologias existentes em Sistemas Distribuídos. Com a finalidade de englobar as tecnologias, de Trabalho Cooperativo, de Sistemas Distribuídos e Aplicações Multimídia, está sendo desenvolvida no Departamento de Computação da UFSCar, dentro do programa Recope/Finep[2] - Computação de Alto Desempenho, a plataforma JAMP (*Java Architecture for Media Processing*) [FER97, FER98].

[1]Financiado pela CAPES
[2]Processo n°3609/96

A plataforma JAMP é um ambiente de programação distribuída baseada em Java/RMI (*Remote Method Invocation*), composta por vários *frameworks* de serviços. O RMI integra a tecnologia de Sistemas Distribuídos diretamente com a linguagem Java. A invocação remota de métodos de Java/RMI permite a comunicação de objetos, através da chamada de métodos, em máquinas diferentes em uma aplicação distribuída.

Segundo [TAN97], as aplicações distribuídas têm crescido em importância e utilização sob a influência de vários fatores, dentre os quais a evolução de tecnologias de hardware e software. Dentre as características que contribuem para o crescente interesse nesta área, tem-se: a *Flexibilidade*, que espalha a carga de trabalho por todas as máquinas disponíveis ao longo da rede; o *Crescimento Incremental*, que permite o poder computacional crescer conforme a necessidade; a *Confiabilidade*, que garante o funcionamento do sistema como um todo, quando uma de suas máquinas deixa de funcionar; e a *Distribuição Inerente*, uma vez que algumas aplicações envolvem máquinas separadas fisicamente.

Em Sistemas Distribuídos problemas como ambientes heterogêneos, gerenciamento de configuração, e monitoramento de rede são críticos. A orientação a objeto é uma solução, onde a modelagem de um Sistema Distribuído é feita através de uma coleção de objetos interagindo entre si, proporcionando uma melhor forma para integração no processamento das informações distribuídas.

Como a linguagem Java é orientada a objetos, suporta as exigências dos objetos distribuídos e é voltada ao desenvolvimento de aplicações para WEB, esta é a linguagem utilizada neste trabalho. Tanto a plataforma JAMP quanto as aplicações que são desenvolvidas reutilizando os *frameworks* da plataforma JAMP são multiplataforma, pois utilizam Java/RMI como linguagem de implementação.

Este trabalho tem como objetivo apresentar um *framework* para o desenvolvimento de aplicações para suportar o Trabalho Cooperativo na plataforma JAMP e apresentar a validação deste *framework*, através de um protótipo de um editor gráfico cooperativo.

Os detalhes das tecnologias que foram unidas para atingir o objetivo deste trabalho são apresentadas nas próximas seções. A seção 2 apresenta uma visão geral sobre Trabalho Cooperativo. A seção 3 apresenta os *Frameworks*. A seção 4 a Plataforma JAMP. A seção 5 um *Framework* para suportar ao Trabalho Cooperativo na Plataforma JAMP e finalmente, na seção 6, são apresentadas as conclusões deste trabalho.

2. Visão Geral sobre Trabalho Cooperativo

Em meados dos anos 70, a crescente preocupação em aumentar a produtividade das organizações deu origem a uma área de pesquisa chamada Automação de Escritórios. Os primeiros esforços nesta área buscavam integrar e transformar aplicações mono-usuário, como processadores de texto e planilhas eletrônicas, de forma a permitirem o acesso simultâneo de um grupo de usuários.

As pesquisas realizadas na área de Automação de Escritórios observaram a necessidade de realizar estudos sobre o comportamento dos grupos ao desempenhar uma atividade. Estes estudos serviram como base para o desenvolvimento de ambientes mais adequados para suportar o Trabalho Cooperativo. Para gerar estes ambientes, técnicos aliaram-se a profissionais da área de humanas, como por exemplo, sociólogos, psicólogos, antropólogos e educadores. Após isto, o termo Automação de Escritórios deu origem a sigla CSCW (*Computer Supported Cooperative Work*) [CSC98].

O termo CSCW costuma ser usado como sinônimo de *groupware*, porém alguns autores identificam uma tendência diferenciada no emprego destes termos. Enquanto CSCW é usado para designar a pesquisa na área do trabalho em grupo e como os computadores podem apoiá-lo, *groupware*

tem sido usado para designar a tecnologia (hardware e/ou software) gerada pela pesquisa em CSCW [ELL91].

A construção de um suporte para as aplicações *groupware* depende de três áreas [ELL91]:

- *a Comunicação:* possibilita aos usuários o compartilhamento de informações entre diferentes tecnologias computacionais. Segundo [ROG92], os problemas de comunicação, principalmente a não confiabilidade, podem causar enormes transtornos a interação cooperativa, em alguns casos inviabilizando a aplicação;
- *a Colaboração*: permite os usuários trabalharem de forma simultânea, como por exemplo na edição de um mesmo documento. As dificuldades desta área chave está no controle da janela de edição comum de instâncias distintas da aplicação. Através desta janela, os usuários atuam modificando o documento e podem visualizar as alterações dos demais usuários sobre o mesmo documento; e
- *a Coordenação*: garante a consistência global dos objetos quando os usuários estão editando o mesmo documento. Mecanismos de garantia de consistência dos objetos devem ser usados para os casos de tentativas de acesso simultâneo sobre um mesmo objeto.

2.1 Requisitos e preocupações no desenvolvimento de aplicações cooperativas

Fatores tais como a facilidade de uso, confiabilidade, legibilidade, robustez, compatibilidade, reusabilidade, são requisitos e preocupações de uma aplicação cooperativa. Entretanto, de acordo com [PIS99], devido as características bem definidas do Trabalho Cooperativo, como o dinamismo das aplicações e necessidade de conectividade, surgem outras preocupações mais específicas, que serão abordadas a seguir.

Ambiente de trabalho dinâmico

A organização de um trabalho em um grupo não é estática, desenvolvendo-se de acordo com as mudanças nos relacionamentos dos indivíduos, nos processos de trabalho e modificações do ambiente. Um grupo de trabalho pode ter variações no número de indivíduos que o compõe, nas responsabilidades de cada indivíduo dentro do grupo e no fluxo dos processos de trabalho.

Uma aplicação cooperativa deve ser flexível o suficiente para acompanhar as mudanças dos processos de um grupo. Além de estar preparada para estas modificações, uma aplicação cooperativa deve funcionar como um agente de mudanças no ambiente de trabalho, fazendo com que processos de cooperação sejam mais incentivados.

Meios de informação

Para que em um grupo trabalhe de forma cooperativa, é necessário uma grande troca de informações entre seus integrantes. Essas informações podem ser resultados de discussões, pesquisas, projetos, ou mesmo documentos administrativos, que podem de alguma forma ser requisitados pelos participantes do grupo.

É necessário então, que estas informações estejam armazenadas de forma que a recuperação, atualização, acesso e qualquer outra atividade que as envolva, possa se dar da maneira mais eficaz possível. A utilização de ferramentas computacionais oferece facilidades para estas atividades. Com o desenvolvimento da multimídia, a informação no ambiente de trabalho pode ser armazenada da

maneira que melhor a represente, através de textos, gráficos, sons, imagens ou animação. Os sistemas de *groupware* devem ter ferramentas para a manipulação destas formas de representação.

A seguir é apresentada a tecnologia a ser utilizada para construir aplicações que suportem o Trabalho Cooperativo, os *Frameworks*.

3.*Frameworks*

As aplicações orientadas a objetos são construídas por classes que colaboram entre si, através da troca de mensagens, para realizar as tarefas do sistema. Através do paradigma orientado a objetos é possível reutilizar uma aplicação ou parte desta, somente redefinindo o comportamento de algumas subclasses. A partir da reutilização de uma aplicação existente pode-se obter diferentes aplicações, reutilizando tanto o código como o projeto geral desta aplicação. Analisando-se o comportamento esperado das aplicações é possível detectar que existe uma parte da atuação destas que é comum e que pode ser generalizado. Assim, pode ser criada uma nova classe, superclasse das anteriores, que contenha todo o código comum a ambas, enquanto as originais implementam a parte específica do comportamento abstrato.

As classes abstratas representam conceitos genéricos relativos a uma família de objetos relacionados. Cada objeto representa um caso particular da abstração e será representado por uma subclasse concreta, que fornecerá uma variante específica do comportamento abstrato definido na classe abstrata. Desta maneira, as classes abstratas trabalham como um modelo para as suas subclasses. Assim sendo, um projeto constituído por classes abstratas funciona como um "molde" para aplicações. Um projeto constituído por classes abstratas é denominado um *framework* de aplicação orientado a objetos ou simplesmente *framework* [ARA97, LEV99, BOS97].

3.1 Vantagens de utilizar *frameworks*

A utilização de *frameworks* proporciona vantagens ao desenvolvimento de aplicações, como:

- Reduz a manutenção [LEV99, SOU98]. Devido a herança, quando no *framework* um erro é corrigido ou uma nova característica é adicionada, os benefícios destas mudanças são disponibilizadas rapidamente para as classes derivadas.
- Reduz o tempo de desenvolvimento de novas aplicações [LEV99, SOU98]. Como a estrutura do programa já está especificada, o fluxo de execução definido, a maioria das classes necessárias já estão codificadas e testadas, então o desenvolvedor não precisa descobrir novas classes, projetar interfaces, etc. Basicamente, ele somente necessita de redefinir alguns métodos específicos de determinadas classes.
- Aumenta a compatibilidade entre as aplicações [NIK95]. As aplicações que compartilham o mesmo *framework* trabalham de forma semelhante. Então os usuários operam as aplicações de maneira mais fácil, pois as interfaces são as mesmas ou similares.
- Permite o desenvolvimento de aplicações mais complexas [SOU98]. A criação de *frameworks* baseados em outros *frameworks* permite o desenvolvimento de aplicações cada vez mais complexas.
- Um *framework* bem projetado facilita a sua utilização e aumenta o seu reuso. Entretanto, o desenvolvimento de *frameworks* exige um longo tempo de projeto.

A seguir é apresenta a plataforma JAMP, nesta é realizada a união de *Frameworks*, Trabalho Cooperativo, Aplicações Multimídia e Sistemas Distribuídos.

4. A Plataforma JAMP

A plataforma JAMP é um ambiente de programação, que oferece mecanismos que embutem vários formatos de mídias, serviços e codificações dentro das aplicações, assim como seu processamento e recuperação. A plataforma JAMP possui uma arquitetura distribuída, que é formada por vários servidores, um *Broker* e um conjunto de *frameworks* usados pelas aplicações multimídia distribuídas.

A Figura 1 mostra aplicações multimídia distribuídas e um conjunto de *frameworks* já disponíveis na plataforma JAMP.

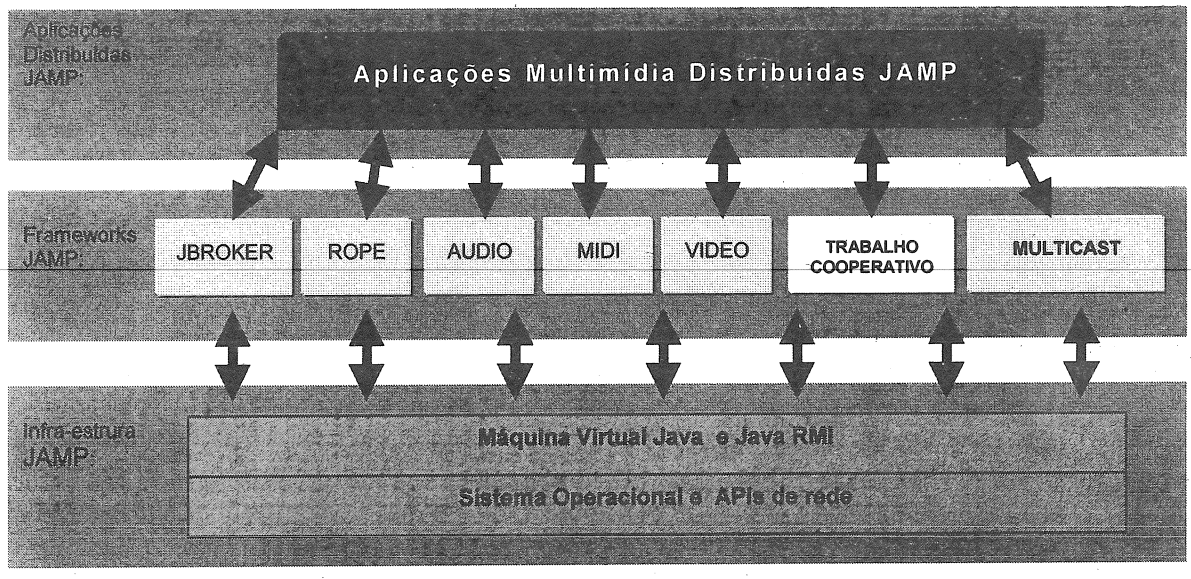

Figura 1 – Plataforma JAMP

Aplicações Multimídia Distribuídas JAMP: As aplicações multimídia distribuídas JAMP podem usar todos os *frameworks* disponíveis na plataforma. Os serviços dos *frameworks* são embutidos nas aplicações através de sua compilação. As aplicações podem ser: *Aplicações Java*, que são executadas pelo interpretador Java; ou *Applets*, que são executadas pelos *Browsers* ou *Applet viewers*.

***Frameworks* JAMP:** Atualmente a camada *frameworks* dispõe dos seguintes *frameworks*: *Rope, Audio, MIDI, Video, Trabalho Cooperativo* e o de *Multicast*. Estes *frameworks* dispõem de APIs [FER98] que são usadas pelas aplicações para incorporar as necessidades de processamento de mídias, Trabalho Cooperativo e de comunicação multicast. O *framework* de Trabalho Cooperativo é o objetivo proposto neste trabalho, este servirá como molde para a construção de aplicações cooperativas. Nesta camada nota-se a flexibilidade que a plataforma JAMP oferece às aplicações para WEB, pois novos *frameworks* podem serem criados conforme a necessidade, ou seja, a plataforma não está restrita apenas aos *frameworks* citados. Com o surgimento de novos serviços para WEB, novos *frameworks* podem ser criados, estendendo a plataforma com estes novos serviços.

O *JBroker* é um *framework* que oferece importantes funcionalidades para a plataforma. Contém uma base de dados dos servidores disponíveis no momento e permite que os clientes encontrem os

servidores distribuídos (objetos remotos) na rede. Um cliente JAMP pode invocar um método de um servidor depois da invocação com sucesso, por referência remota no *JBroker*. O processo de invocação, que inicia com o servidor sendo registrado no *JBroker* para uso direto da aplicação, é chamado de *Trading Process*, conforme apresenta a Figura 2.

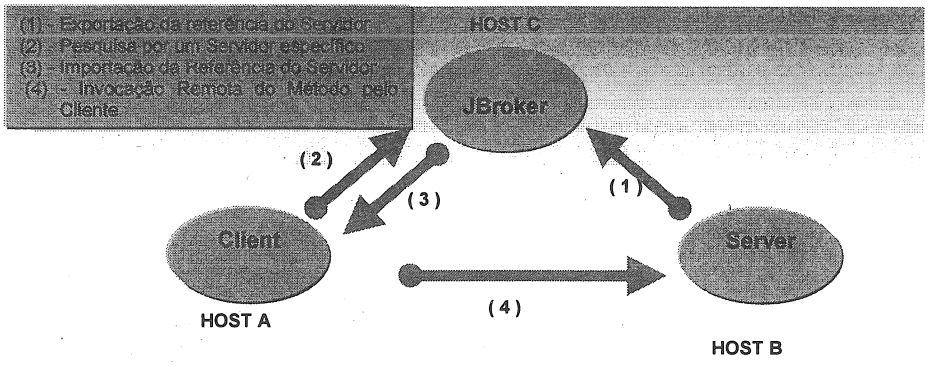

Figura 2 – Trading Process

Infra-estrutura da plataforma JAMP: Implementada em Java, a plataforma JAMP permite invocar métodos de objetos remotos como se fossem objetos locais. Estes podem estar em diferentes arquiteturas de hardware e diferentes JVM (*Java Virtual Machine*). O TCP/IP é o responsável pela comunicação. A Figura 3 mostra uma visão geral da camada RMI Cliente/Servidor.

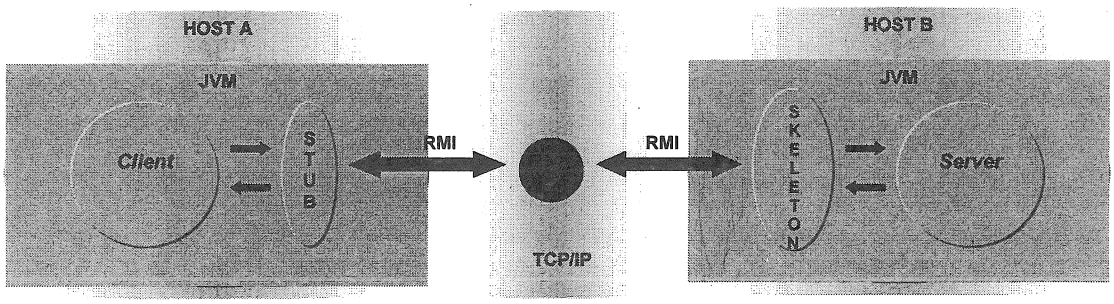

Figura 3 – Visão Geral do mecanismo RMI na plataforma JAMP

A Arquitetura Distribuída JAMP: é formada pelo *Java Broker*, vários servidores e por um conjunto de *frameworks* usados pela aplicações. A Figura 4 mostra a arquitetura com os servidores JAMP (*Host* B, C e D), *o Java Broker* (*Host* A), e os *frameworks* JAMP embutidos na aplicação multimídia JAMP (*Host* E). As setas *Server Registration* representam o registro dos serviços no *Java Broker* e, as *Client Invocation* mostram os clientes invocando dinamicamente os serviços referenciados pelo *Java Broker*. Após o sucesso da invocação, os clientes podem acessar diretamente por RMI. As setas RMI identificam todas as possíveis conexões RMI.

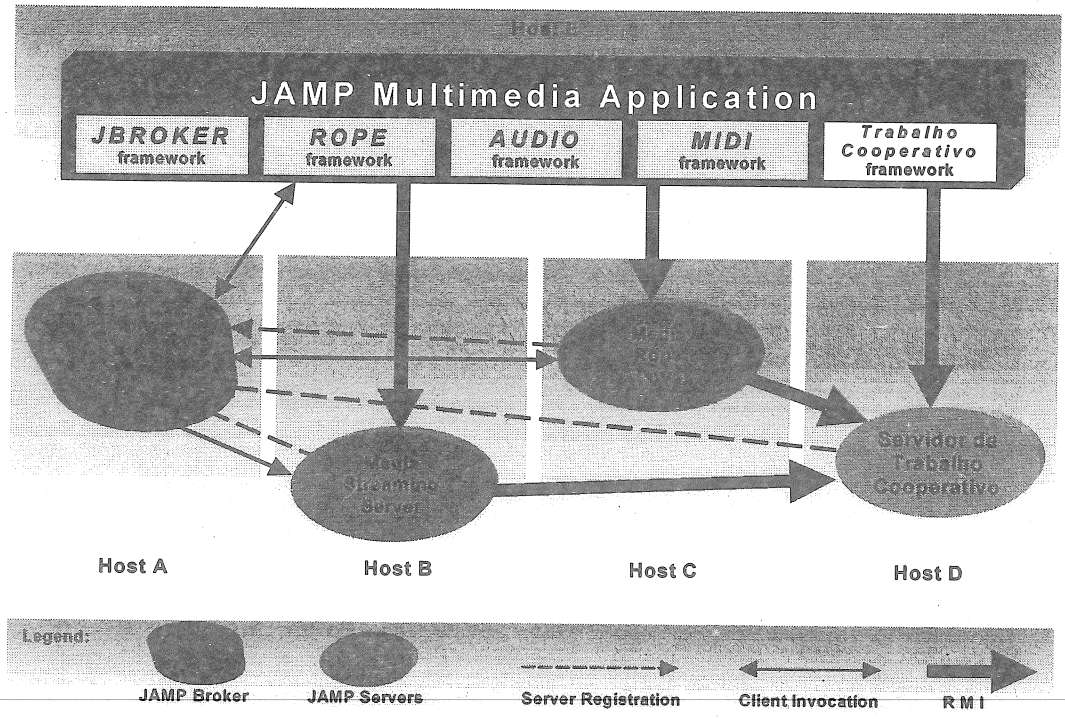

Figura 4 - Arquitetura Distribuída JAMP

A seguir é apresentado o *framework* de suporte ao Trabalho Cooperativo e a validação deste, através da construção de um editor gráfico cooperativo.

5.Um *Framework* de Suporte ao Trabalho Cooperativo na Plataforma JAMP

Após análise das aplicações de Trabalho Cooperativo, chegou-se ao resultado que algumas partes gerais deste tipo de aplicação podem ser generalizadas, dentre estas tem-se:

- o acesso compartilhado à objetos (controle de concorrência) – segundo [MIT96, FRI95a, FRI95b, GRE94] no ambiente de Trabalho cooperativo tem-se a probalidade de conflitos, que é gerada pelo fato dos usuários realizarem operações em objetos compartilhados de forma síncrona e em computadores diferentes; e
- a percepção (*Awareness*) – de acordo com [MIT96, GRE95] a percepção serve como uma forma de coordenação das atividades e ajuda a evitar trabalho conflitante e redundante. Exemplos de mecanismos de percepção são: o uso de cores distintas para cada usuários, canais de comunicação e históricos de mudanças.

Em cada uma dessas partes são identificadas diversas alternativas, as quais podem ser implementadas através de um *framework*. Assim, o *framework* de Trabalho Cooperativo é composto por diversos métodos de controle de concorrência e mecanismos de percepção. Quando os desenvolvedores forem construir suas aplicações, estes podem reutilizar as funcionalidades deste *framework*, assim como, de todos os outros da plataforma JAMP, de acordo com os requisitos das aplicações a serem construídas. Portanto, a plataforma JAMP oferece aos desenvolvedores um ambiente que permite a criação de diversas aplicações de Trabalho Cooperativo, como por exemplo: Sistemas de Apoio a Decisão, Sistemas de Conferência, Editores de Texto Cooperativos e Editores Gráficos Cooperativos.

A Figura 5 mostra algumas classes e métodos do *framework* de Trabalho Cooperativo.

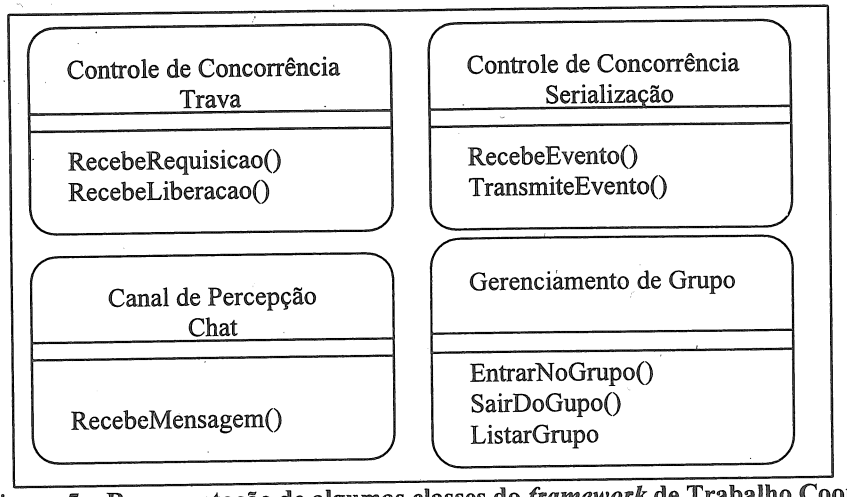

Figura 5 – Representação de algumas classes do *framework* de Trabalho Cooperativo

Para avaliar o *framework* de Trabalho Cooperativo e a plataforma JAMP como suporte computacional, um editor gráfico cooperativo está em desenvolvimento. Este editor manipula documentos gráficos que são compostos por objetos geométricos como: linhas, retângulos, polígonos, círculos, elipses e outros, e além disso, possivelmente figuras gráficas pré-existentes. A Figura 6 mostra o protótipo da interface deste editor.

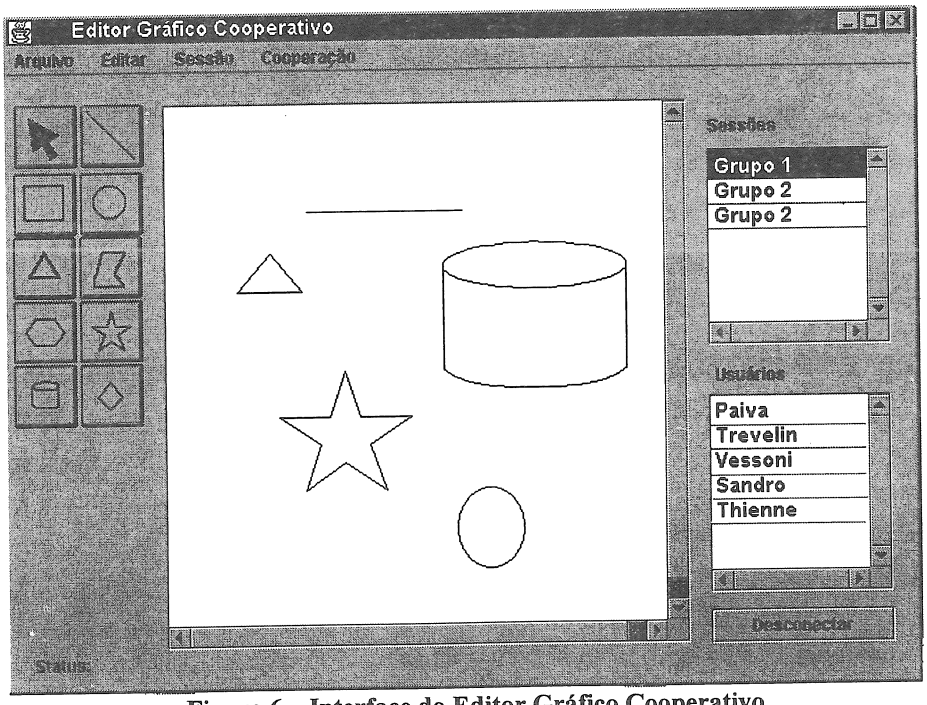

Figura 6 – Interface do Editor Gráfico Cooperativo

Neste editor simples, através de menus e botões de interface gráfica, o usuário pode manipular os desenhos (incluir, apagar, modificar atributos), manipular os documentos (criar um novo, ler e salvar um documento em edição), entrar ou sair de uma sessão, visualizar quais são os usuários que estão cooperando e trocar mensagens. O editor é uma *Aplicação Java,* que pode ser executado pelo

Interpretador Java ou na forma de *Applets*, que podem ser executadas pelos *Browsers* de navegação Web ou pelo *Appletviewer*.

Para evitar o problema de inconsistência dos objetos que estão sendo compartilhados na janela de edição comum, o editor faz reuso de um método de controle de concorrência disponível no *framework* de Trabalho Cooperativo da plataforma JAMP, o método *trava não-otimista* [GRE94].

Este método primeiramente trava (bloqueia) o acesso dos outros usuários sobre o objeto requisitado, de forma a impedir a execução de atualizações concorrentes sobre o mesmo objeto, a manipulação do objeto não é permitida antes de receber a resposta de trava concedida ou não. Quando um acesso a um objeto é requisitado, é verificado se ninguém mais possui a trava, então o acesso é garantido, caso contrário é negado.

Como o editor trabalha com granularidade média, ou seja, não trabalha com pixel e nem com objetos muito grandes, isto justifica o uso deste método *trava*. A forma *não-otimista* é justificada devido este trabalho prever o uso de redes e computadores de alto desempenho, consequentemente não há problemas de latência neste ambiente.

Seguindo a arquitetura da plataforma JAMP, a distribuição dos objetos deste editor está especificada conforme exemplifica a Figura 7. Esta arquitetura criada pela distribuição de objetos é composta por diversos *Hosts*, sendo que no *Host* A está ativo o *JBroker*, no *Host* B o Servidor de Trabalho Cooperativo e no *Host* E o Editor Gráfico Cooperativo. Por meio do protocolo TCP/IP estes se comunicam de forma remota. O Servidor de Trabalho Cooperativo é um repositório central para manipulação e armazenamento de dados, e é responsável pela disponibilização dos métodos necessários para o funcionamento do sistemas (como por exemplo, o controle de concorrência e mecanismos de percepção). O *framework JBroker* da plataforma JAMP, disponibiliza para as aplicações, neste caso para o editor, o tratamento de exceções remotas, a localização inicial dos métodos do Servidor e a criação de sessões.

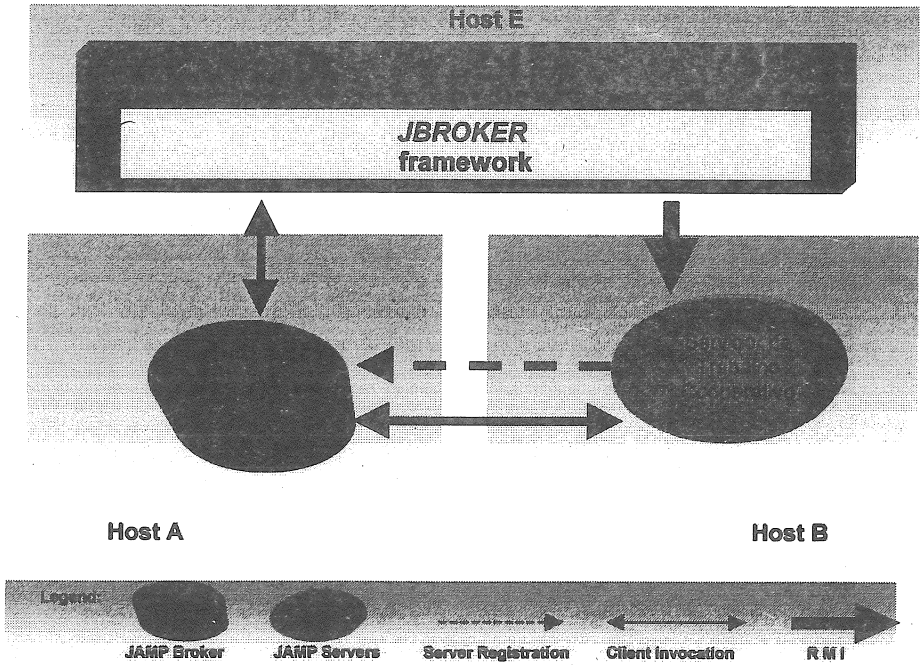

Figura 7- Arquitetura do Editor Gráfico Cooperativo

Na Figura 7, as aplicações invocam dinamicamente os serviços remotos do Servidor através do *Java Broker* (*client invocation*). Depois da primeira invocação com sucesso, a aplicação acessa o Servidor diretamente via RMI. No editor, o Servidor de Trabalho Cooperativo (*Host* B) ao ser disponibilizado será registrado no *Java Broker* (*Host* A) para que seja oferecido como um serviço à aplicação editor (*Host* E). A aplicação que utilizar o *framework JBroker* ao executar pela primeira vez o acesso remoto, receberá através do *JBroke*r a localização do Servidor e estabelecerá a conexão RMI.

Para implementar o Editor utiliza-se a interface *InterServ,* a classe cliente *Editor* e a classe servidor *ServidorEditor*. A Figura 8 mostra a interface *InterServ*, nela são definidos os métodos disponibilizados pela classe *ServidorEditor* para a classe cliente *Editor*.

```
public interface InterServ extends Tradeable {
    // Controle de Concorrencia - Trava
   public void RequisitaTrava(Objeto obj) throws RemoteException ;
   public void LiberaTrava(Objeto obj) throws RemoteException;
                . . .
}
```

Figura 8 – Implementação da interface *InterServ*

A Figura 9 mostra a implementação da classe servidor do Editor, a classe a *ServidorEditor*. Esta classe implementa a interface remota a ser distribuída (*InterServ*) e a interface de erros remotos (*JBErros*). A classe *ServidorEditor* quando invocada criará uma nova seção no *Java Broker* e exportará os métodos para serem invocados pelas classes clientes (como por exemplo, os métodos *RequisitaTrava* e o *LiberaTrava*).

```
import javabroker.*;
import cooperativo.*;
     . . .
public class ServidorEditor extends java.rmi.server.UnicastRemoteObject implements
InterServ,   JBErrors{
 . . .
public static void main( String args[] ) {
 . . .
 { cria uma nova seção, exporta o servidor Editor para o JBroker, trata os erros de
exportação}
  ...
// Controle de Concorrencia – Trava
  public void RequisitaTrava(Objeto obj) throws RemoteException {
       . . . {reutiliza os métodos do framework de Trabalho Cooperativo -
RecebeRequisicao}
 }
 public void LiberaTrava(Objeto obj) throws RemoteException {
       . . . {reutiliza os métodos do framework de Trabalho Cooperativo -
RecebeLiberacao}
 }
. . .
}
```

Figura 9 – Implementação da classe *ServidorEditor*

A Figura 10 mostra a classe cliente *Editor*, inicialmente são declaradas duas variáveis, a variável *Servidor*, utilizada para acessar o objeto remoto e a variável *session*, utilizada para criar uma nova seção no *Java Broker*. Quando o construtor for executado, ele criará uma nova seção no servidor (*host* onde está localizado o *Java Broker*), tendo o cuidado de tratar todos os erros possíveis. Dentre os diversos métodos da classe tem-se o *SelecionaObjeto* e o *LiberaObjeto*, estes métodos são os responsáveis pela requisição da obtenção e liberação da trava sobre um determinado objeto.

```
public class Editor extends java.applet.Applet implements  JBErrors {
  InterServ Servidor ;
  public JBSession session;
                                        . . .
public Editor() throws RemoteException {
   { cria uma nova seção no servidor}
          . . .

void SelecionaObjeto(java.awt.event.MouseEvent event){
     { verifica qual objeto foi selecionado }
      . . .
   Servidor. RequisitaTrava (obj);  {requisita a trava}
      . . .
  }
}

void LiberaObjeto(Event event) {
      . . .
          { libera a trava do objeto }
      Servidor. LiberaTrava (obj);
      . . .
   }
}
```

Figura 10 – Implementação da classe *Editor*

6. Conclusão

Este trabalho apresentou uma proposta da união de Trabalho Cooperativo, Aplicações Multimídia e Sistemas Distribuídos, dando origem a um *framework* para suporte ao Trabalho Cooperativo na plataforma JAMP e a um editor gráfico cooperativo para testar este *framework*.

A união dessas tecnologias aplicadas na plataforma JAMP gerou um ambiente de programação distribuído simples, adequado para o desenvolvimento de aplicações que suportem o Trabalho Cooperativo na WEB. Principalmente devido ao fato da plataforma JAMP suportar Java, que é uma linguagem que permite a programação distribuída, e além disso, atende os requisitos necessários para o desenvolvimento de aplicações WEB.

Assim, as áreas chaves de suporte que um ambiente deve oferecer às aplicações de Trabalho Cooperativo foram apresentadas neste trabalho: a *Comunicação*, através das funcionalidades da plataforma JAMP e da linguagem Java/RMI; a *Colaboração* e a *Coordenação* por meio da reutilização dos *frameworks* desta plataforma.

7. Referências Bibliográficas

[ARA97]ARAUJO, R.M. & DIAS, M. S. & BORGES, M.R.S. **A Framework for the Classification of Computer Supported Collaborative Design Approaches**. III CYTED-RITOS International Workshop on Groupware. Madrid Spain. October.1997

[BOS97]BOSCH, J. & MOLIN, P. & M. MATTSSON, M. & BENGTSSON, P. **Object-Oriented Frameworks: Problems & Experiences**. 1997. http://bilbo.ide.hkr.se:8080/~michaelm/fwpages/fwbibl.html. Consultado em 18/01/1999.

[CSC98]**CSCW, Groupware & Internet**. http://www.cos.ufrj.br/~renata/cscw/sumario.htm. Consultado em 25/11/1998.

[ELL91]ELLIS, C.A. & GIBBS, S.J. & REIN, G.L. **Groupware : Some Issues and Experiences.** Communications of the ACM. vol.34. nº1. pp. 39-58. January 1991.

[FER97]FERREIRA, M. M. & TREVELIN, L. C. **A Platform for Developing Distributed Multimedia Application .** Project report. DC.UFSCar.1997.

[FER98]FERREIRA, M. M. & TREVELIN, L. C. **The Java Broker System: Concepts &Java Programming Guide.** Project report. DC.UFSCar.1998.

[FRI95a]FRITZKE JR, U. **Projeto e Implementação de um Suporte para Aplicações Cooperativas do Tipo Editor Distribuído**. Dissertação de Mestrado. Departamento de Pós-Graduação Engenharia Elétrica. Universidade Federal de Santa Catarina. Florianópolis. Novembro de 1995.

[FRI95b]FRITZKE Jr, U. & FARINES, J. **Um Suporte para Aplicações de Trabalho Cooperativo do Tipo Editor Distribuído**. 13° Simpósio Brasileiro de Redes de Computadores. Universidade Federal de Minas Gerais. Departamento de Ciência da Computação. Belo Horizonte. Brasil. de 22 a 26 de maio de 1995.

[GRE94]GREENBERG, S. & MARWOOD, D. **Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface**. ACM Conference on Computer Supported Cooperative Work (CSCW).Chapel Hill. North Carolina. USA. pp. 207-217.1994.

[GRE95]GREENBERG, S. & HAYNE, S. & RADA, R. **Groupware for Real-Time Drawing:** A designer's Guide. McGRAW-HILL.1995.

[LEV99]**LEVERAGING Object – Oriented Frameworks.** http://www.ibm.com/java/education/ooleveraging/. Consultado em 27/01/1999.

[MIT96]MITCHELL,A. **Communication and Shared Understanding in Collaborative Writing**. Thesis at Toronto University, Canada. 1996. http://www.dgp.toronto.edu/people/alex/thesis/. Consultado em 10/12/1998.

[NIK95]NIKLAS, L. & NIKLASSON, A . **Development of Object-Oriented Frameworks.** Master Thesis. Department of Communication Systems. Lund University. 1995. http://www.tts.lth.se/Personal/bjornr/Papers/OOFW.ps. Consultado em 08/02/1999.

[PIS99]PISTORI, J. **A tecnologia CSCW no processo ensino-aprendizagem.** http://www.geocities.com/CapeCanaveral/2106/cscw.htm. Consultado em 17/01/1999.

[ROG92]ROGERS, Y. **Ghosts in the Network: Distributed Troubleshooting in a Shared Working Environment**. Proc. 4th Conference on Computer Supported Cooperative Work. pp. 346-355.1992.

[SOU98]SOUZA, C. **Um Framework para Editores de Diagramas Cooperativos baseados em Anotações**. Dissertação de Mestrado. Universidade Estadual de Campinas. 12 de outubro de 1998.

[TAN97]Tanenbaum, Andrew S. **Redes De Computadores.** 3ª edição .ed.Campus.1997.