SID - Sistema Interativo Distribuído

Proposta de projeto

Sistemas de Objetos Distribuídos

Prof.: Fabio Kon
IME/USP – Maio 2002

Aluno: **Jorge F. Del Teglia**

Objetivos deste documento

O objetivo principal deste documento é apresentar generalidades do projeto antes de ser implementado tanto assim como decisões de arquitetura já em analise. Não é o objetivo deste documento resolver todos os problemas que possam surgir durante o desenvolvimento nem apresentar uma solução final e fechada.

As idéias e conceitos apresentados neste trabalho serviram de guia para a construção final do segundo projeto.

Objetivos do projeto

Este projeto é uma extensão radical ao trabalho numero um realizado no curso. As extensões realizadas são de dois tipos diferentes:

1. oferecimento de serviços para permitir a interação remota de diversos usuários, utilizando uma ferramenta muito mais que textual: neste caso, servindo de metáfora uma "lousa virtual" (*virtual blackboard*);

2. modificações arquiteturais no projeto anterior visando generalizar a estrutura e permitir a adição de novos elementos funcionais.

Funcionalidades e características

O objetivo deste projeto é abrangente e sem limites claramente definidos, com o intuito de continuar a adicionar funcionalidades nele, mesmo após da entrega do trabalho pedido neste curso. Mesmo assim, alguns requisitos surgem como possibilidades interessantes e são enumerados a continuação:

- o sistema permitirá a coexistência de dois tipos de sessões de comunicação diferentes (esta coexistência só é possível a traves da mudança de arquitetura no sistema original):
    - sessão padrão de bate-papo (*chat)* como no modelo anterior;
    - sessão de *Hyperchat*, nome dado ao tipo de sessão que permite a interação tanto textual como gráfica.
- nas sessões de bate-papo, o usuário origem da comunicação poderá invitar a um conjunto de outros usuários cadastrados a participar na interação;
- os usuários dentro de uma sessão de *Hyperchat* poderão, simultaneamente, realizar operações gráficas de diverso tipo na "lousa". Estas operações serão percebidas em tempo real por todos os outros participantes da sessão. Por exemplo, se os usuários A, B e C estão numa sessão de *Hyperchat* e o usuário A desenha um retângulo na tela, os usuários B e C poderão ver na sua tela o mesmo desenho. Variadas operações, como serem desenhar retângulos, círculos, linhas, pontos, agrupar figuras, desagrupar figuras, mover, copiar e apagar elementos, estarão disponíveis;
- se a velocidade da conexão o permite, os usuários poderão ativar a opção de acompanhamento do mouse dos outros usuários, vendo na sua tela as posições dos ponteiros dos mouses de cada um dos outros usuários na sessão, com um indicador do nome do usuário nela. Esta opção, porem provavelmente com alto consumo de recursos (de comunicação principalmente), é interessante porque incrementa potencialmente a interatividade oferecida aos usuários.
- também é de interesse que a arquitetura seja o suficientemente aberta para permitir a incorporação de funcionalidades interativas distribuídas adicionais. Se a implementação mínima é concluída, são dois os tipos de interação suportados: interação textual e interação gráfica. Em aras de avançar na interatividade de um sistema distribuído, novos tipos de interações poderiam ser adicionados, como reprodução de sons onomatopaicos e *emoticons* (clássicos nos sistemas de bate-papo). Comunicação falada, envio e recebimento de arquivos, compartilhamento de arquivos em tempo real, reprodução simultânea de filmes e músicas (supondo,

claro, que as características de infra-estrutura da rede sejam suficientes). Em resumo, os médios interativos envolveriam principalmente três áreas:

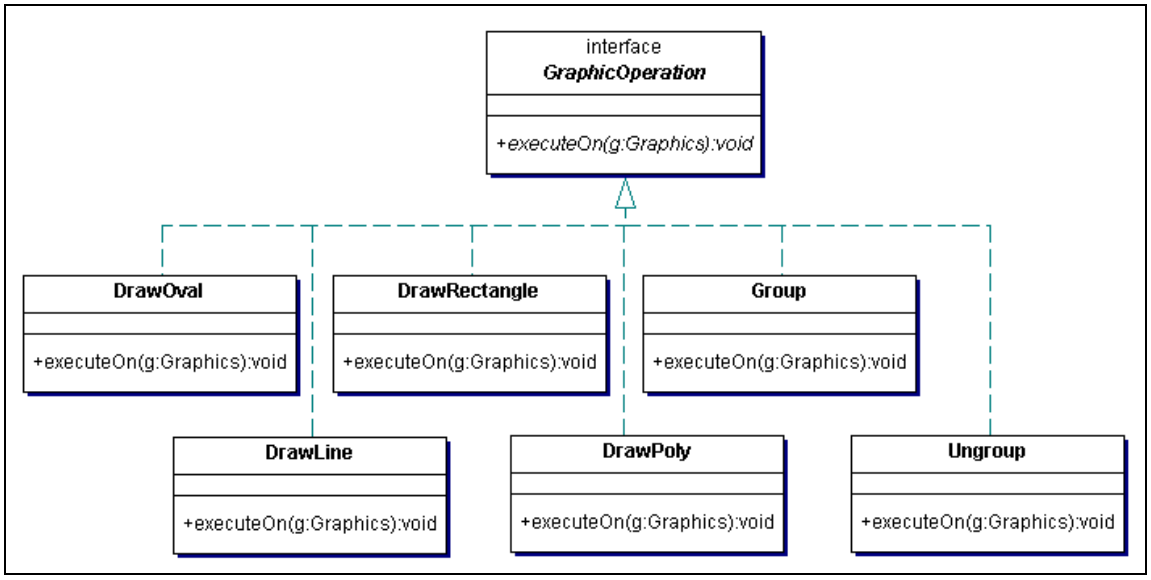
- **Visual:** textos, gráficos, *emoticons*, filmes;
- **Auditiva:** sons onomatopéicos, língua falada, interface de sintese de voz para usuários com incapacidades vocais;
- **Informação empacotada:** arquivos variados.

Arquitetura geral do sistema

É para destacar que a arquitetura adotada terá uma influência determinante na hora de permitir tamanha quantidade de serviços. Por isso, esta arquitetura deveria ser suficientemente aberta para permitir continuas extensões mas também suficientemente fechada como para permitir implementar o sistema antes de definir claramente quais outras funcionalidades interativas são desejadas e como serão implementadas.

A base arquitetural escolhida para uma primeira iteração (o projeto real poderia-se propagar por muitas) está baseada no encapsulamento de funcionalidades a serem executadas distribuidamente em objetos, tendo estes um supertipo em comum que permita um tratamento básico das funcionalidades mas gerais. Desta forma, o tratamento das operações será mais homogêneo e dessa forma diminuirá a complexidade de tratamento de operações sensivelmente diferentes. Este conceito está muito bem estudado no padrão de projeto *Command* [1].

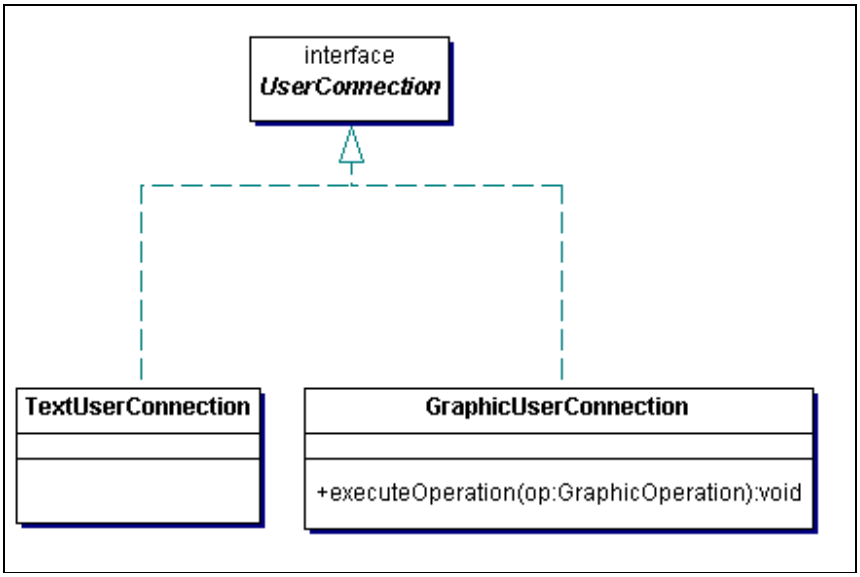
Como na versão inicial, existirão duas aplicações a serem rodadas, uma no servidor, e outra em cada cliente, mas adotando a arquitetura associada ao padrão *Command* conseguiremos um desacoplamento significativo entre eles. Muitas possíveis atualizações no servidor não requereram atualizações nos clientes. No pior caso, o cliente desatualizado não saberá como apresentar o resultado de executar certo comando, mas os comandos anteriores seguiram funcionando corretamente.



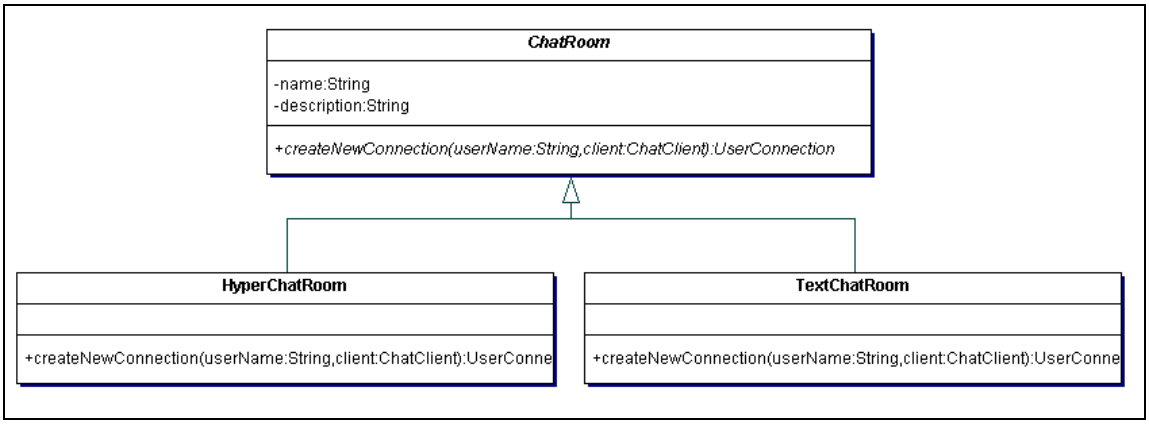
Como mostra a figura, diferentes classes implementarão a interface principal e cada uma delas proverá uma implementação acorde ao problema a ser resolvido. Porem o exemplo está focado na parte gráfica (parte principal deste trabalho), o *núcleo* arquitetural deixa espaço para extensões em varias direções.

Paralelamente as diferentes operações é interessante possibilitar restrições no tipo de operações que um cliente pode fazer através da conexão ganha com o servidor (por exemplo, pensemos que existam diferentes perfis, alguns deles com mais permissões e autorizados a utilizar operações que tem um consumo de recursos maior). Estes tipos diferentes de sessões seriam, em principio, diferentes implementações de UserConnection.

Um corte de uma versão simplificada do diagrama de classes principal do sistema, mostra que, a diferencia da versão inicial, agora teremos uma especialização das conexões, sendo uma delas para as sessões padrões e a outra para as sessões de *Hyerchat*.



Para gerenciar os diferentes tipos de conexões grupalmente, encapsulamos a criação de conexões de certo tipo em objetos. Isto é, mantemos também diferentes tipos de salas de bate-papo, sendo cada uma delas responsável pela apropriada criação de conexões e outorgando uniformidade no tratamento delas desde os módulos que utilizam seus serviços (como ser, por exemplo, a interface gráfica de usuário (IGU)). Esta micro-arquitetura está baseada no padrão de projeto *Factory Method* [1].



O volume de trabalho para a implementação do projeto completo é significativo, mas o objetivo inicial de preparar a arquitetura para sua possível extensão é o primeiro grande passo. A utilização extensiva de padrões de projeto (*design patterns*) ajuda muito na comunicabilidade das idéias arquiteturais escolhidas para este projeto, sendo os elementos mencionados neste trabalho ferramentas básicas (por ser a base, mas não por serem simples) para construir aplicações distribuídas robustas e manuteníveis.

Bibliografia

1. Erich Gamma et al: **Design Patterns: elements of reusable object-oriented software** – Addison Wesley, 1995.