ÍNDICE

1 SUMÁRIO

O projecto consiste na implementação de uma aplicação de suporte a reuniões remotas electrónicas. Caracteriza-se por criar versão electrónica de um dispositivo muito usado nas reuniões convencionais, o *flipchart*, complementado com um meio que permite o diálogo embora que na forma de texto, tipo *Internet Relay Chat* (IRC). O componente *flipchart* é um dispositivo que permite uma maior expressividade tornando mais evidente as relações entre a informação criada, enquanto que o IRC permite o diálogo entre as pessoas para troca de informações ocasionais ou de apoio à interpretação.
Conjuntamente têm o objectivo de reduzir o tempo dispendido durante uma reunião através do aumento da produtividade, da criatividade e redução nas deslocações dos intervenientes.

2 INTRODUÇÃO

As pessoas que têm experiência na área das reuniões, usam cada vez mais meios de acelerar a reunião e reduzir os custos em todo o processo que envolve a realização de uma reunião. Alguns estudos sobre reuniões concluem que nas reuniões existe pouca produtividade, pouca organização nas ideias expostas e dificuldade em atingir consenso. As razões para tais resultados resumem-se à falta de meios de suporte ao processo de uma reunião assim como o desgaste provocado pelas deslocações frequentes. Por tais razões, os sistemas de suporte à decisão em grupo e reuniões remotas electrónicas têm recebido cada vez uma maior adesão. No entanto, existem aspectos que necessitam de um maior desenvolvimento, pois algumas das vantagens das reuniões face-a-face convertem-se em desvantagens. Estas desvantagens têm de ser minimizadas através da eficiência do suporte fornecido e com a qualidade da informação gerada.

A área em estudo centra-se na fase "durante a reunião", cobrindo as fases de geração, esclarecimento e elaboração de ideias, nas situações em que os participantes numa reunião se encontram geograficamente distribuídos. Assim, uma forma de resolver a falta de meios de suporte durante uma reunião remota electrónica consiste na implementação de um meio onde a criação e alteração de informação seja feita de uma forma rápida e eficiente. O processo de criação deve aproximar-se tanto quanto possível da forma convencional e deve haver um meio de despiste das ambiguidades geradas.

O projecto convergiu para a implementação de uma ferramenta de suporte às reuniões remotas através de um *flipchart* electrónico que é complementado com um IRC. A escolha de um IRC deve-se ao facto de não serem utilizados meios áudio ou vídeo.

Para a implementação definiu-se uma arquitectura cliente-servidor onde a comunicação é estabelecida via *Internet*. Foi usado o Java™ como linguagem de programação.

Com base no *flipchart* electrónico foi desenvolvida um linguagem visual própria para reuniões remotas.Com esta aproximação ao modelo mental, utilizado na tarefa convencional, reduziu-se a inércia inicial à utilização dos novos meios. A introdução de um meio de esclarecimento, como pode ser um IRC, reduz a dificuldade em fazer uma reunião remota decorrente da ausência do diálogo e esclarecimento face-a-face.

Uma maior liberdade na geração de informação fará igualmente os utilizadores esquecerem algumas das dificuldades de fazer uma reunião remota electrónica.

3 CONCEITOS BÁSICOS

O conceito de reunião é muito lato e é passível de ter várias interpretações. Isso porque pode existir vários processos de elaboração, desenvolvimento e tipos de reuniões e assim surge a necessidade de definir o ambiente em que este projecto se enquadra. No entanto, outras definições surgem da necessidade de realçar algumas diferenças entre reuniões remotas e face-a-face..

3.1 REUNIÕES

As reuniões são uma necessidade constante de qualquer tipo de organização. Numa sociedade, os diversos tipos de relações entre pessoas fazem com que a maioria dos assuntos sejam resolvidos após concordância do grupo como um todo.
O processo que envolve uma reunião subdivide-se em três fases: Pré-reunião, durante a reunião e pós-reunião, para um conhecimento aprofundado sugere-se a consulta de [4].

3.1.1 REUNIÕES CONVENCIONAIS

Corresponde ao tipo de reunião mais comum. Caracteriza-se por decorrer à mesma hora e no mesmo local, onde um grupo de pessoas discute os vários temas em questão e após concordância ou votação são determinadas as conclusões. O material utilizado pode ser variado, desde bloco de notas a quadros tipo *whiteboard* ou *flipchart*, entre outros dispositivos. Pode ter ou não suporte de um facilitador.

3.1.2 REUNIÕES ELECTRÓNICAS

Todo o processo de uma reunião e a elaboração das conclusões nem sempre são executadas de uma forma eficiente e eficaz. Daí surge a necessidade de criar ferramentas para tornar as reuniões mais produtivas.
A utilização destas ferramentas tem como objectivo resolver alguns aspectos ainda pouco desenvolvidos nas reuniões, como informação insuficiente, inadequada definição dos objectivos, insuficiência de planeamento, inadequada preparação dos participantes, a condução inadequada da reunião pelo coordenador (facilitador), o tempo gasto na revisão das reuniões anteriores e a atenção excessiva a questões menores
De modo geral, nas reuniões electrónicas os participantes interagem através de uma ou mais aplicações num computador. As salas especialmente preparadas para suportar reuniões electrónicas estão equipadas com um computador para cada participante que está ligado em rede aos restantes participantes e a outras tecnologias de suporte a reuniões, por exemplo *whiteboards*, projectores de vídeo, entre outros. Tem usualmente a intervenção de um facilitador.

3.2 FACILITADOR

Juntar pessoas para trabalhar eficazmente e em conjunto é um dos maiores desafios de liderança nas organizações actuais. Os líderes têm de alcançar a forma ideal para cativar uma participação activa e comprometer-se a dar suporte às equipas cujos membros tenham outras actividades e/ou preocupações em paralelo que desviem as suas atenções. A facilitação é pois a arte de juntar as pessoas e pô-las a trabalhar com eficácia [15].
A acção do facilitador, no contexto de uma reunião, é visto como um conjunto de funções e actividades desempenhadas antes, durante e após a reunião para ajudar o grupo a determinar as conclusões dada uma situação inicial. O facilitador caracteriza-se por ter uma acção que permite atingir facilmente uma conclusão [4].
Nas reuniões é dada uma especial atenção ao papel do facilitador humano que tem como tarefa coordenar todas as fases da reunião, estando também atento a aspectos que, nas reuniões convencionais face-a-face são, muitas vezes, resolvidas sem qualquer tipo de intervenção deste género. O facilitador cria a ordem de trabalhos e manda executar as diversas ferramentas da forma que tinha planeado, enquanto os participantes respondem às solicitações das ferramentas escolhidas. Já foram feitos muitos estudos sobre as insuficiências nas técnicas de intervenção à disposição do facilitador, incluindo o limitado suporte à facilitação remota, que foi detectado num estudo feito em [10].

O objectivo do facilitador é pois a ajuda e melhoramento do desempenho dos participantes, a identificação dos problemas e o encaminhamento dos intervenientes no sentido de uma solução para o problema apresentado [2].
Existem duas possíveis intervenções do facilitador. Este pode acumular as tarefas inerentes à função de facilitador, as actividades de participante ou, apenas intervir como facilitador.
Até o guru da gestão, Peter Drucker reconhece o papel fundamental do facilitador na citação:
"Organizações do futuro serão cada vez mais baseadas em informação, sendo organizadas não como as empresas industriais mas como as orquestras, hospitais ou universidades. As organizações serão compostas basicamente por especialistas que orientam o seu próprio desempenho através do *feedback* dos outros. Neste contexto desenvolver-se-ão as organizações baseadas em equipas, nas quais os especialistas se encontrarão em reuniões, tomando conjuntamente as decisões sem intervenção dos gestores profissionais " [6].

3.3 Reunião electrónica remota e Facilitação remota de reuniões electrónicas

Devido à necessidade de reduzir os custos relacionados com as reuniões, a reunião electrónica remota e a facilitação remota de reuniões electrónicas têm sido alvo de alguma atenção através de novas teorias e estudos das ferramentas existentes [2][3][4][5][8][9][14]. No caso específico da reunião electrónica remota, esta permite uma optimização bastante visível quer na componente tempo quer na componente custo de todo o processo que envolve uma reunião convencional [4]. A necessidade de facultar um maior apoio ao facilitador remoto surge do facto deste ter a seu cargo várias tarefas em paralelo, tendo de estar atento a uma série de pormenores enquanto está a processar outras tarefas, não podendo interagir visualmente e/ou auditivamente com os participantes. O facilitador tem assim o seu desempenho comprometido visto estar a orientar e a apaziguar situações remotamente. Algumas das vantagens das reuniões face-a-face tornam-se em desvantagens nas reuniões remotas e a necessidade de haver mecanismos eficientes que façam, em parte, ou dêem suporte ao trabalho do facilitador, surge naturalmente.

3.4 Sistema de suporte a reuniões electrónicas

Consiste num conjunto de ferramentas desenvolvidas para ajudar todas as tarefas que os participantes e especialmente o facilitador tem de executar remotamente [4].

O projecto enquadra-se num ambiente de uma reunião remota electrónica e orientada por um facilitador, que pode ser também participante da reunião. A reunião decorre sem a inclusão de canais auditivos e vídeo, restringindo a interacção ao texto e gráficos.

4 TÉCNICAS

4.1 SITUAÇÃO ACTUAL

Uma descrição da situação actual permite o esclarecimento dos aspectos prementes no âmbito deste projecto. Importa realçar que devido ao âmbito deste projecto essa descrição centra-se na fase durante a reunião.

4.1.1 DESCRIÇÃO

Os "processos entre pessoas" são a essência de qualquer organização. A sua acção pode ser sentida na forma como as pessoas planeiam, inovam, conduzem reuniões, tomam decisões, comunicam as estratégias, coordenam projectos, redesenham programas/calendarizações, resolvem problemas e melhoram a qualidade [15].
Em qualquer organização, seja qual for o seu tamanho, a comunicação é a chave para a implementação eficaz de novas aproximações e permite atingir resultados mais positivos [15].
A maioria dos sistemas de suporte a decisões em grupo (SSDG) comerciais estão direccionados para as reuniões electrónicas face-a-face com suporte para a facilitação face-a-face. Apesar da possível utilização destes sistemas para a realização de reuniões remotas, estes oferecem um grau reduzido de suporte à facilitação remota.
Estudos feitos com intenção de analisar a eficiência da utilização de uma linguagem visual numa reunião concluem que [16]:

- 64% das pessoas decidem-se imediatamente após uma apresentação visual da informação;
- Reduz o tempo de duração da reunião em 24 %;
- Promove o consenso do grupo;
- É persuasiva;
- As apresentações usando linguagem visual causam uma melhor impressão;
- A resolução de problemas e compreensão de documentos é facilitada.

O ambiente analisado corresponde a reuniões que decorrem no mesmo local e à mesma hora e os resultados apresentados são por comparação às apresentações orais. Mas pode-se concluir que a linguagem visual pode ser muito poderosa em vários aspectos independentemente do ambiente em que esta é utilizada, ainda mais quando a componente oral e auditiva não existe.

Uma revisão sobre pesquisas experimentais de reuniões concluem que:

- As reuniões consomem uma grande parte do tempo e esforço das organizações;
- A maioria das reuniões são tidas como sendo extremamente pouco produtivas em termos de eficácia utilizando o tempo dos participantes e para a obtenção eficaz dos objectivos propostos da reunião.

As áreas mais problemáticas que levam a reuniões pouco produtivas são:

- Desenho de reunião pobre → desorganização e falta de preparação;
- Focus pobre → pouco conhecimento do assunto a tratar;
- Falta de conclusão → reuniões que não acabam, resultados incompletos;
- Processo fraco → discussão superficial das alternativas.

Estes resultados devem-se à falta de criação e utilização de linhas directrizes na reunião assim como na falta execução de procedimentos mais produtivos. Este último caso acontece pelo insuficiente treino, inexperiência e a resistência à mudança.
Constatou-se que o sistema de suporte a reuniões pode ter um impacto positivo nos processos de grupo, resultando em tarefas e relações optimizadas [8].

4.1.2 PROBLEMAS

Existem algumas limitações na concretização de reuniões remotas que podem ser ultrapassadas com a utilização de técnicas avançadas de facilitação. Problemas relacionados com o *feedback* de informação, a desmotivação dos participantes habituados a utilizar a argumentação e persuasão, a necessidade da identificação, reduzindo a facilidade de expressividade e espontaneidade.

Nas reuniões remotas electrónicas surge a necessidade de reduzir as desvantagens que surgem da reunião não ser face-a-face. Essa necessidade gera tarefas extra que o facilitador tem de desempenhar para uma fluência normal da reunião. Para tal será necessário criar mecanismos que levem os intervenientes a gerarem informação de uma forma rápida e o mais parecida possível aos meios utilizados numa reunião convencional e terem acesso a toda informação de uma forma contextualizada. Actualmente estas técnicas ainda não estão suficientemente desenvolvidas nas ferramentas disponíveis.
Nas reuniões electrónicas remotas os problemas multiplicam-se dado que existem pormenores nas relações humanas face-a-face que são difíceis de transpor para as reuniões. Há uma necessidade de desenvolver pequenos mecanismos que facilitam a tarefa do facilitador.

4.1.3 MOTIVAÇÃO

As razões que motivam ao desenvolvimento de uma nova forma de gerar, esclarecer e discutir ideias numa reunião remota electrónica são:

- A utilização de uma linguagem visual mais semelhante ao modelo conceptual da acção permite a redução da dificuldade em compreender a aplicação no seu todo. A utilização desta metodologia de desenvolvimento poderá ser encontrada e aprofundada em [15]. Assim criação de meios que se aproximam do modelo mental do processo de uma reunião leva a uma menor rejeição desse meio e a uma melhor compreensão da forma de interacção com este. Isto é obtido com a elaboração uma linguagem visual que, através das metáforas, faz com que o utilizador reduza a adaptação apenas à forma de interagir com a aplicação. A interacção é diferente quer por razões do contexto em que é utilizado quer por razões técnicas de implementação.
- A criatividade e produtividade aumenta drasticamente se todos conseguirem perceber qual é o assunto em discussão [15].
- Conseguir mostrar as ideias de uma forma rápida e com a possibilidade de alteração ou até redefinição dá uma maior segurança a quem as expõe.
- A promoção do consenso, com a possibilidade de esclarecimentos, permite que o interveniente compreenda melhor as ideias expostas.

O conhecimento da identidade de qualquer interveniente na reunião pode levar a exposições menos espontâneas e criativas por parte destes. Isso deve-se ao facto do conhecimento da identidade poder ter um efeito inibidor, muito devido à imagem criada perante a audiência. Assim, a possibilidade de anonimato surge como uma ajuda à expressividade do interveniente.

4.1.4 OBJECTIVOS

Este projecto pretende resolver o problema da geração, exposição e organização de ideias nas reuniões electrónicas remotas. Para isso pretende-se criar mecanismos que facilitem a exposição textual das ideias e principalmente permitir uma gestão, organização e relacionamento dessa informação.
No desenvolvimento da aplicação, o objectivo é facultar a interacção entre todos os intervenientes, participantes e facilitador, de uma forma simples, optimizando a troca de informação e disponibilizando mecanismos de armazenamento e organização da informação.
A aplicação pretende dar a possibilidade de comunicação directa entre os participantes, semelhante aos *IRC*′s comerciais, bem como métodos para dar suporte às actividades que surgem durante a reunião.

4.2 SISTEMA IDEALIZADO

Na descrição da solução idealizada são tidos em conta os problemas apresentados na descrição da situação actual.
A descrição do sistema idealizado só é feita após a definição do tipo de utilizador e da tecnologia a usar na implementação.

4.2.1 CARACTERIZAÇÃO DO TIPO DE UTILIZADOR

O utilizador típico caracteriza-se por ser uma pessoa com pouca experiência informática. Este necessita de saber utilizar teclado, rato, ter alguma experiência na utilização de interfaces gráficas e estar familiarizado com a *Internet*.

4.2.2 REQUISITOS DE SOFTWARE

Sendo a ferramenta direccionada para reuniões remotas, a *world wide web* (*www*), ou simplesmente *Internet,* é escolhida como meio de comunicação dado a sua fácil acessibilidade.
A aplicação é desenvolvida em *Java™* na versão *1.2.2,* desenvolvido pela *Sun Microsystems*, e através do *JBuilder Fundation,* versão 3.5, para a criação das interfaces gráficas. No desenvolvimento das interfaces é usado o pacote *swing* do *Java Fundation Classes* (JFC), parte integrante do *Java™ 1.2.2*. Este é um pacote vital pois contém componentes poderosos para a construção de interfaces gráficas.
Para a implementação da aplicação é necessário a utilização de um servidor que permita a gestão da comunicação. O servidor de *http* utilizado é o *vqServer©*, versão 1.9.39, desenvolvido por SteveShering da vqSoft.
São ainda necessárias ferramentas complementares ao desenvolvimento, sendo estas um editor de texto simples e um qualquer *browser* para efeitos de testes. O editor de texto é do tipo *notepad*, para o desenvolvimento da página onde o carregamento da aplicação é feito. Os *browsers* podem ser o *Netscape Communicator©* , versão 4.72, da *Netscape Communications Corporation* ou o *Microsoft Internet Explorer©*, versão 5.0, da *Microsoft Corporation*, ou o *Hot-Java Browser*. Os dois primeiros *browsers* estão largamente difundidos, muitas vezes através de Sistemas Operativos que permitem um acesso a informação que é independente da plataforma em que correm.
A escolha da linguagem de programação *Java™* deveu-se basicamente à portabilidade e robustez das aplicações desenvolvidas.
A utilização da linguagem de programação Java™ é:

Simples, orientada a objectos e facilmente compreendida por quem já tem alguma experiência em programação, nomeadamente C/C++;
Robusta e segura e;
Interpretada e dinâmica.

Permite:

Multithreading;
Uma arquitectura neutra e portável e;
Elevado desempenho.

Vantagens de usar Java:

Evita dependências entre plataformas e;
Distribui o software mais facilmente → Permite que nos *applets* possam ser carregados novas classes "*on the fly*" sem recompilar todo o programa.

Ao evitar as dependências entre plataformas, a aplicação desenvolvida pode ser lançada em qualquer *browser* que a suporte.
Justifica-se também a escolha do *Java™* pela razão da aplicação usar a *Internet* como meio de acesso e comunicação. A robustez é atingida pelo facto desta linguagem ter sido desenvolvida para explorar o máximo das potencialidades de um meio como a *Internet*, facilitando o desenvolvimento de aplicações para esse ambiente.
O *Java™* suporta ainda *multithreading,* que permite a execução de diversas tarefas em simultâneo. Assim toda a comunicação com o servidor não é feita através de um único *thread* no *servlet* porque com muitos clientes iria haver estrangulamento muito cedo, seria como se estivesse apenas um único

cliente a ser servido de cada vez. Através do *socket* o servidor cria um *thread* para cada pedido o que faz com que as respostas sejam muito mais rápidas devido à sincronização ser mais fina.

A comunicação entre cliente e servidor é feita usando s*ockets* e *HTML* via servidor de *http*.

4.2.2.1 Restrições de Software e Hardware

Para a utilização da aplicação é necessário um *browser* que suporte aplicações *em Java™ 1.2.2* ou no qual possa ser usado *Java Plug-In*, que é distribuído com o *Java Runtime Environment* .
Pode ser utilizado qualquer Sistema Operativo e uma plataforma que suporte o *Java Virtual Machine.*

4.2.2.2 Restrições externas

O ambiente de uma reunião remota electrónica, mesmo orientada por um facilitador leva a uma forte dependência das tecnologias utilizadas. Consequentemente surgem alguns aspectos inerentes à robustez da aplicação desenvolvida tais como atrasos na troca de informação, conhecidos por tempo de latência de comunicação numa rede, relacionados com a distância entre os nós, intensidade do tráfego e até qualidade do meio de comunicação. Tudo tem ainda um maior relevo quando o ambiente de implementação é a *Internet*, que não se trata de uma rede com tais aspectos controláveis.
Há que ter em conta o facto de nem todas as versões dos *browsers* estarem optimizados para as novas versões de *Java™*, que surgem mais rapidamente que as versões dos *browsers*.

4.2.3 MODELO CONCEPTUAL

O esboço do modelo a implementar é apresentado na **Figura 1** seguido da exposição das decisões tomadas.

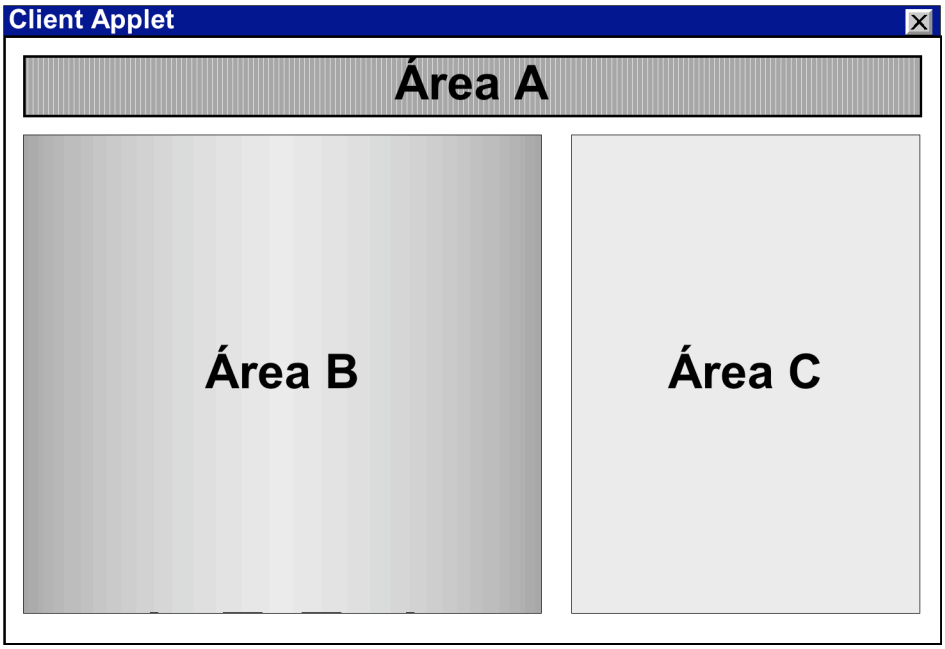

Figura 1 - Esboço do modelo da aplicação.

O modelo da aplicação consiste em três áreas distintas, com a qual o interveniente vai interactuar.
A **Área A** é a zona onde se encontra um *toolbar* de botões para o acesso à informação referente ao decorrer da reunião e para o controlo da própria sessão.
A **Área B** é a zona do *flipchart*. No *flipchart* existe um *toolbar* com um conjunto de botões que permite a acção do utilizador através da introdução de informação quer textual quer gráfica. Note-se que não está representado graficamente na **Figura 1** para manter a simplicidade do modelo e realçar a distinção entre as áreas representadas.

A **Área C** é a zona do IRC. O IRC tem o funcionamento comum aos IRC´s comercializados e permite a troca livre de mensagens entre intervenientes, com a emissão e visualização destas. Qualquer interveniente pode e deve usar esta área para complementar e esclarecer as suas ideias ou posições.

4.2.4 DESCRIÇÃO DO MODELO

Este projecto consiste na implementação de um modelo de comunicação para uma aplicação de suporte a reuniões remotas electrónicas. Essa aplicação funciona no contexto da *www* e permite que um grupo de pessoas possa fazer uma reunião sem estarem presentes fisicamente.
A aplicação a implementar tem a sua utilidade durante a reunião. A fase durante a reunião não é mais que a interacção que utiliza um conjunto de recursos (pessoas e tecnologias) para transformar o estado actual do grupo (problema) num estado desejado (resultado) para cada tópico da lista de tópicos que é a agenda. Para um dado tópico da agenda, que já fora previamente definida na fase de pré-reunião, há as sub-fases de gerar, discutir, organizar e comunicar as ideias. Isto repete-se para cada tópico da agenda, como pode verificar na **Figura 2**. Após a obtenção de consenso ou da votação de ideias é fechada a reunião. Todo este modelo é desenvolvido em [4].

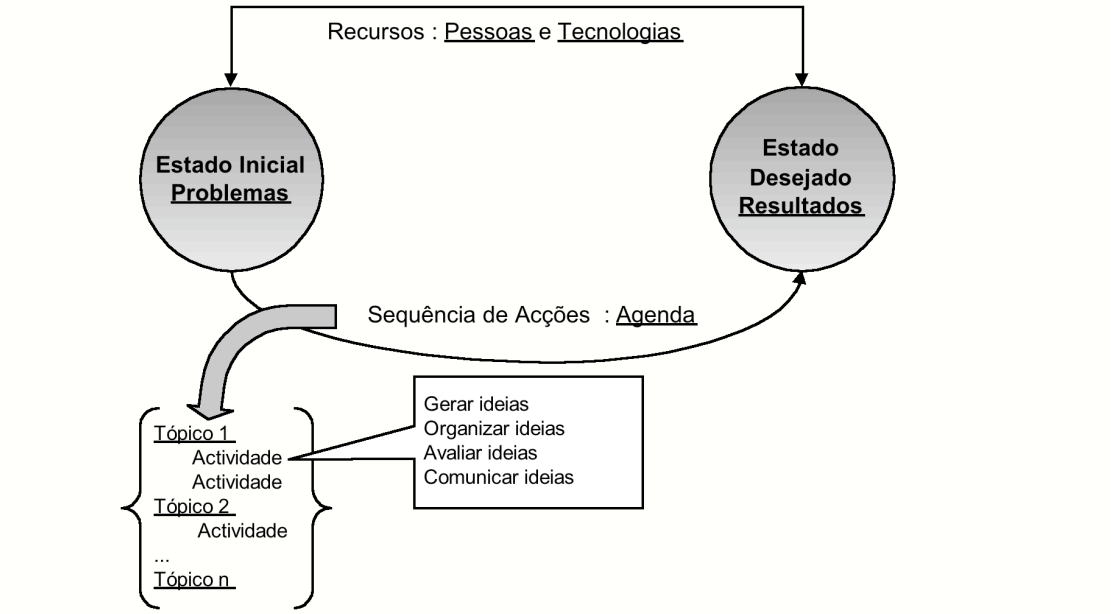

Figura 2 - Modelo de uma reunião [4].

Para as sub-fases referidas anteriormente, como gerar, discutir, organizar e comunicar ideias, vamos criar meios de tornar as tarefas mais simples de executar. Todo o processo, visto decorrer em ambiente remoto, visa o estabelecimento de um canal de comunicação comum a todos os intervenientes. Existe duas formas paralelas de comunicar, uma é a edição de mensagens de um para todos, concretizada através de um estação de *IRC*, que passaremos a chamar *SessionDialog*, a outra consiste elaboração de informação gráfica através de um painel que simula um *flipchart*, que passaremos a chamar *SessionChart*.
A reunião decorre num ambiente informal onde a verdadeira identificação de cada interveniente depende exclusivamente deste, no entanto tem de ter uma identificação para que possa ser referenciado. Assim o anonimato dos intervenientes existe se estes assim o desejarem.

O acesso à lista de participantes é facultado, permitindo que se possa visualizar os nomes, fictícios ou não e a função que desempenham na reunião.
Como já foi referido, a aplicação tem a sua utilização na fase durante a reunião e como tal os intervenientes têm acesso aos tópicos da agenda previamente definida pelo facilitador na fase pré-reunião.
Cada interveniente pode entrar na reunião atrasado, sair e voltar a entrar que no momento da entrada tem sempre acesso à informação gerada até então. Permite uma integração rápida por parte do interveniente que acede tardiamente. Este tem acesso imediato a toda a informação gerada até então.
A recolha de informação de uma forma rápida dá origem à funcionalidade de copiar texto da área do *SessionDialog* para a área do *SessionChart*.

4.2.4.1 Esquema do *SessionDialog*

O *SessionDialog* é constituído por duas áreas, representado na **Figura 3,** onde a Área C2 representa a zona para a introdução de dados e a Área C1 representa a zona para a zona de visualização de todas as mensagens trocadas até então.

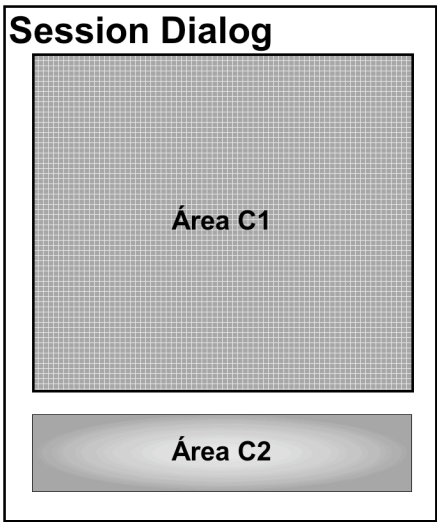

Figura 3 - Modelo conceptual do *SessionDialog*.

4.2.4.2 Funcionamento

O funcionamento do *SessionDialog* é idêntico aos sistemas *IRC* comercializados onde a mensagem é gerada por um emissor que após o envio, todos podem visualizar na área de visualização de mensagens, incluindo o próprio emissor. O envio de mensagens não tem qualquer tipo de controlo permitindo que todos possam enviar ao mesmo tempo.

4.2.4.3 Motivos

A forma de interacção com uma ferramenta do tipo *IRC*, para além de já estar razoavelmente difundida, é também uma mais valia numa interface que tende a ser burocrática e onde as ideias possam ter várias interpretações. A intenção é introduzir um mecanismo de que se assemelhe ao discurso corrente como é um diálogo face-a-face. No entanto, a única forma de diálogo disponível é o texto. A componente vídeo e áudio não são utilizadas por razões de usabilidade da solução pois tornaria a solução mais lenta, no que diz respeito a tempos de latência. No entanto reconhecemos as potencialidades acrescidas com a integração de tais meios. Assim só é possível "dialogar" textualmente.
O IRC permite que as pessoas entrem na conversa da mesma forma que quando se opina sobre algo entre um grupo de pessoas conhecidas ou não. Esta forma de diálogo tem a vantagem de permitir emitir opiniões sem o emissor ser reconhecido, ou seja, devido à possibilidade do interveniente entrar na reunião como anónimo, é-lhe permitido uma liberdade que não tem nas reuniões convencionais.
O anonimato é pois uma vantagem no que concerne à produtividade pois existe uma menor inibição na exposição das ideias e opiniões.

4.2.4.4 Esquema do *SessionChart*

O *SessionChart* é constituído por três áreas, como representado na **Figura 4**. Na Área B3 encontra-se a zona de desenho, que representa um bloco de folhas. A Área B2 surge como zona com botões e informação relativos ao bloco de folhas. A Área B1 é uma zona constituída por um grupo de botões representativos dos elementos textuais e gráficos que podem ser construídos no bloco de folhas (Área B3).
A informação gráfica gerada pode ser do tipo documentos ou elementos gráficos. O documento permite a introdução de informação em forma de texto e os elementos gráficos permitem a organização e relacionamento entre a informação textual.

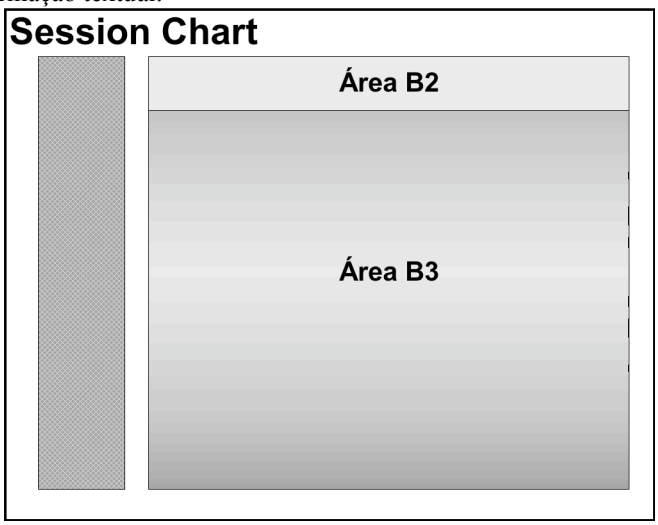

Figura 4 - Modelo conceptual do *SessionChart*.

4.2.4.5 Funcionamento

Nesta área apenas um único interveniente pode estar a criar e/ou alterar a informação do *SessionChart*. Isto para garantir a coerência da informação gerada pois, se assim não fosse, poderia ocorrer a alteração do mesmo elemento de informação por dois ou mais intervenientes, introduzindo o problema de saber qual a informação a armazenar. Assim o interveniente não corre o risco de ser interrompido na exposição da sua ideia, com a vantagem reduzir as ambiguidades que surgem quando a exposição das ideias não está completa.

4.2.4.6 Motivos

A utilização de um *flipchart* cria-se uma linguagem gráfica para as acções desempenhadas num bloco de folhas comum e visível para todos que surge frequentemente nas reuniões convencionais para a exposição e clarificação das ideias.
O *flipchart* permite a descrição simples do que por vezes é complicado verbalmente, através da elaboração de lista de ideias e/ou conclusões de ideias assim como na elaboração de esquemas. As razões da utilização de um *flipchart* prendem-se com a possibilidade de utilização de uma linguagem visual poderosa tendo em conta que nas reuniões remotas em questão, não há intervenção quer auditiva quer visual dos intervenientes, reduzindo a percepção de alguns aspectos inerentes à reunião.

4.3 IMPLEMENTAÇÃO

4.3.1 METODOLOGIA

4.3.1.1 Definição da metodologia de desenvolvimento

Este projecto é desenvolvido usando o modelo de processo de desenvolvimento de software Evolucionário (que é um modelo que permite a incrementação com protótipos). A interface com o utilizador é assim refinada sempre que necessário.

A implementação de uma aplicação em módulos que funcionam de forma independente é sempre uma boa prática de desenho de concepção, que é bem conhecida no desenvolvimento de software, onde as opções de implementação são atrasadas tanto quanto possível no ciclo de vida do produto [3].

Decidimos utilizar este modelo pois, à parte de ser indicado para projectos de longa duração, é o modelo indicado para projectos com muita interacção com o utilizador. Visto este projecto ter uma grande carga de interfaces gráficas, assim a experiência do utilizador complementa os requisitos da aplicação e determina, em parte, a forma como esta será desenvolvida. Há assim necessidade de confrontar o utilizador com um protótipo funcional para que este crie uma opinião e manifeste-a de forma a direccionar o produto final para as expectativas. Neste trabalho há uma procura cessante de requisitos para a aplicação cuja interface permita interagir de uma forma mais eficiente. A alteração da forma como uma dada actividade é executada deve ser feita de modo a que a transição feita pelo utilizador introduza o mínimo de entropia e que este sinta mais benefícios que contrariedades.

Tem a vantagem de permitir o desenvolvimento de partes do produto que estejam dependentes da realização de outras.

Especial atenção é dada à parte da comunicação inerente ao modelo. O estabelecimento de ligação é necessário para que haja fluxo de informação entre os intervenientes.

4.3.1.2 Fases do projecto

As fases:

- Requisitos do Utilizador e
- Requisitos de Software.

Foram desenvolvidas na descrição do sistema idealizado, abordadas respectivamente nos pontos 4.2.1 e 4.2.2 deste texto.

Após a caracterização da situação actual e do sistema idealizado segue-se as fases que descrevem todo o processo de implementação da solução que são:

- Desenho de arquitectura;
- Desenho detalhado - métodos e interfaces de classes
- Codificação

4.3.2 DESENHO DA ARQUITECTURA

A arquitectura é do tipo cliente-servidor onde o meio usado para o estabelecimento da comunicação é a *Internet*, visível na **Figura 5**.

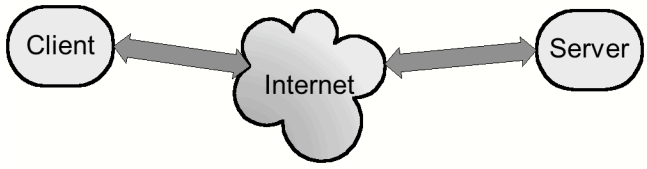

Figura 5 - Comunicação cliente-servidor.

A aplicação é desenvolvida em *Java™*, tendo por um lado o desenvolvimento de um *applet*, correspondente à aplicação visível pelo interveniente ou seja corresponde ao cliente da arquitectura. Um *applet* é uma aplicação em *Java™* que corre num *browser*.
Por outro lado, para que a parte da comunicação seja estabelecida, surge a necessidade de criar *servlets*. Um *servlet* não é mais que uma aplicação em *Java™* que corre sobre o servidor de *http*. Um *servlet* é para um servidor o que é um *applet* para um *browser*. O *servlet* tem a restrição de só poder correr num servidor de *http* implementado em *Java™*.

4.3.2.1 Esquema

A arquitectura centra-se no padrão cliente-servidor. Cada participante (cliente) acede a aplicação através de uma página *HTML* na *Internet* onde, ao seguir o elo *Login* lança uma nova página onde o utilizador escolhe o papel a desempenhar. Essa escolha gera um pedido ao servidor, através do *servlet SessionLogin,* que retorna um *applet* (a aplicação cliente) na mesma página. O *applet* carregado é definido mediante a selecção do papel a desempenhar na reunião (facilitador ou participante). O *applet* estabelece uma ligação com o servidor implementado, o *servlet Router,* que está encarregado de coordenar o processo de comunicação entre os clientes, como está representado na **Figura 6**.

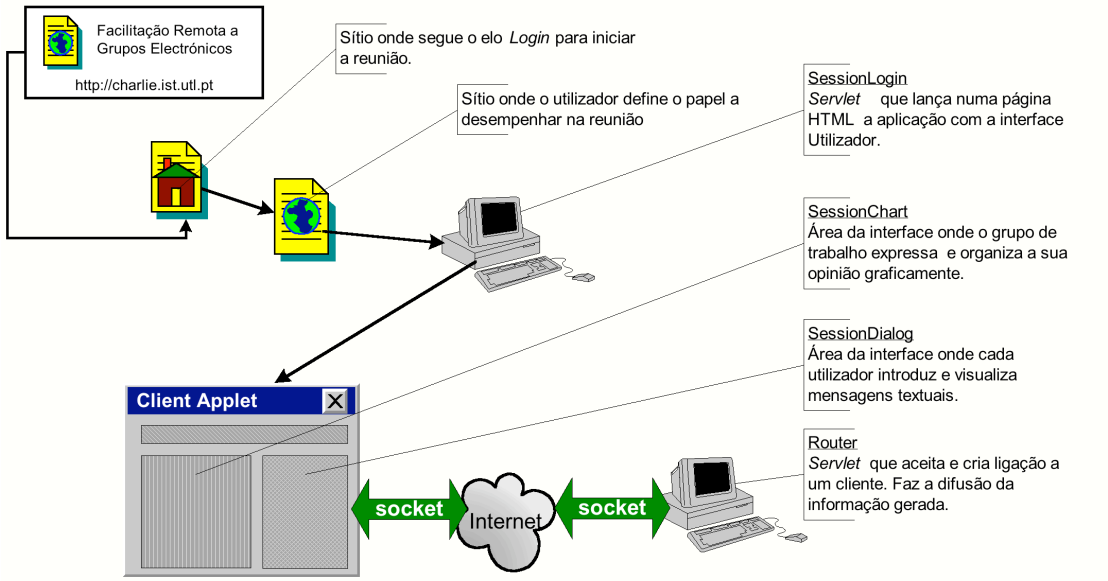

Figura 6 - Comunicação do cliente para o servidor

A comunicação cliente-servidor, estabelecida entre o *applet* cliente e o *servlet Router* usa *sockets* orientados à ligação.

4.3.2.1.1 Funcionalidade

Na comunicação cliente-servidor pode-se distinguir duas situações. Uma acontece na criação de uma sessão, por parte do cliente que estabelece uma ligação entre este e o servidor, através do *servlet SessionLogin*, representado na **Figura 7**. O servlet *SessionLogin* é acedido via servidor de *http*, como já foi descrito anteriormente.

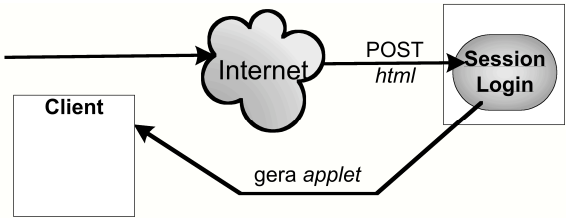

Figura 7 - Pedido de ligação

O pedido de uma ligação por parte do cliente é processado pelo *Router* da forma descrita na **Figura 8**.

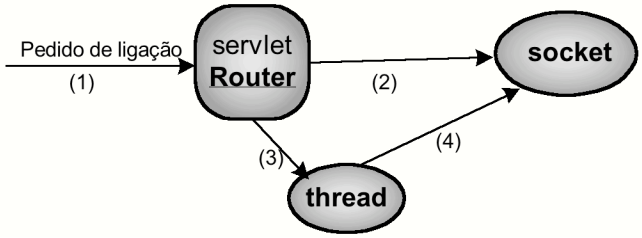

Figura 8 - Pedido de ligação do Cliente ao Servidor.

O cliente ao ser criado, faz um pedido de ligação (1) ao servidor, através do *servlet Router*, cria um *socket* (2) e lança seguidamente um *thread* (3) para a gestão do *socket* (4) estabelecendo-se assim uma ligação de comunicação entre esse cliente e o servidor. Assim, quando cada cliente é criado gera um processo idêntico ao descrito anteriormente. A gestão de pedidos de cada cliente é feita ao nível dos *threads*.
A outra situação consiste na comunicação que é estabelecida entre os clientes. A comunicação é feita através de um *socket* que permitem que cada cliente envie informação para os restantes, passando sempre pelo *Router*, representado na **Figura 9**.

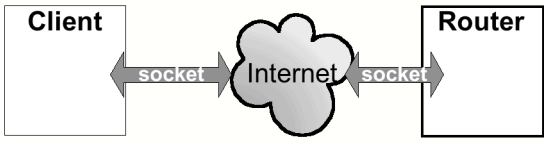

Figura 9 - Ligação entre cliente e servidor

O *servlet Router* que após a sua iniciação, gera o *socket ServerSocket* e lança um *thread* para a sua gestão, coloca o *ServerSocket* em permanente escuta de novas ligações, como é representado na **Figura 10**.

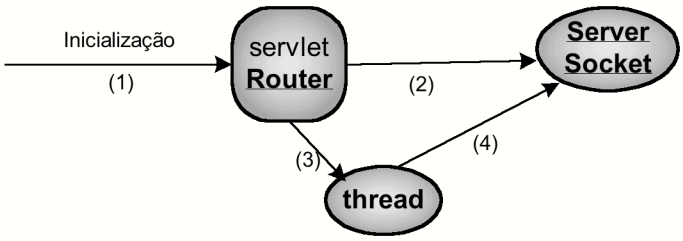

Figura 10 - Inicialização do *servlet Router*.

Como o *ServerSocket* está em permanente escuta de novas ligações, para cada novo pedido de ligação, o *Router* estabelece uma ligação. Assim para os **N** clientes criados existem **N** ligações ao servidor .
Como em qualquer comunicação a informação enviada tem de chegar ao receptor. No entanto a informação não tem um caminho directo ao receptor. A informação que sai do emissor tem de passar pelo servidor. O servidor encarrega-se de difundir a mensagem recebida para todos os intervenientes, até mesmo o emissor, como pode analisar na **Figura 11** com a sequência numérica de setas.

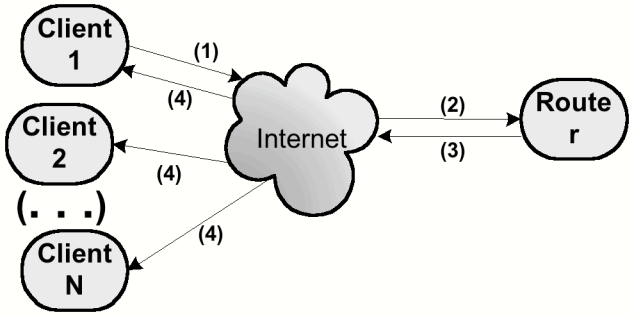

Figura 11 - Fluxo das mensagens.

O cliente envia a informação (1) que segue via *Internet* até ao servidor (2). O servidor faz então a difusão da informação para todos os clientes existentes (3) e cada cliente recebe então a informação, incluindo o cliente emissor.

4.3.2.1.2 Motivos

A opção de uma arquitectura cliente-servidor é inerente à própria concepção do projecto visto que, ao tratar-se de reuniões remotas, a necessidade de uma arquitectura deste tipo surge naturalmente.
A escolha de um *applet* para a parte cliente deve-se ao facto de permitir a independência da plataforma computacional e privilegiar de uma eventual utilização em ambientes heterogéneos [12]. O *applet* não é mais que um programa em *Java*™ que funciona de forma independente. Assim a aplicação poderá ser utilizada a partir de qualquer tipo de plataforma que suporte esta linguagem através de um *browser*. Outra razão é a possibilidade de comunicação entre cliente-servidor e a motivação para criar uma interface graficamente agradável ao utilizador.
Para a parte do servidor poder-se-ia escolher uma aplicação que não necessitasse de um servidor *http*. A aplicação funcionaria de uma forma idêntica ao servidor de *http*, sendo o servidor usado apenas para a gestão de páginas *HTML*. No entanto, se assim fosse, teríamos que construir uma parte gráfica que seria a interface de controlo da aplicação. A criação de parte gráfica para a aplicação servidor torna-se desnecessária se optarmos por um *servlet*, pois ao contrário do *applet*, o *servlet* não permite a construção de uma interface gráfica [13].
Com a opção de um *servlet* a parte de controlo é implementada numa página *HTML* que através de POST de *http* permite o controlo do servidor.
Os *servlets* constituem uma substituição aos *scripts* CGI [13]. A escolha de um *servlet* para servir de acesso ao servidor em vez de um CGI deveu-se à simplicidade, à possibilidade de manter estados, à segurança e à rapidez. Assim é apenas uma aplicação a funcionar em vez de uma série de pequenas aplicações, o que poupa o *overhead* de carregamentos das pequenas aplicações.
Permite a colaboração entre pessoas. Um *servlet* pode receber/responder a vários pedidos concorrentemente e pode sincronizar pedidos. Isto permite que os *servlets* suportem sistemas tipo conferências a tempo real.
Resolvem também o problema de programar o servidor com um API específico para uma plataforma. Estes são então desenvolvidos com o API para *servlets* que é uma extensão do *Java*™ [13].
Os *servlets* podem ser embebidos em diferentes servidores porque o *Application Programming Interface* (API) do *servlet*, com o qual são escritos, não assume nada sobre o ambiente e protocolo do servidor [13].

4.3.3 DESENHO DETALHADO

Visto haver duas partes embora complementares mas distintas da aplicação descrevemo-las de forma separada. Apresentamos também as opções tomadas na implementação da solução proposta.

4.3.3.1 Servidor

O servidor consiste numa aplicação do tipo *servlet* que está a correr numa máquina remota. O *servlet SessionLogin* quando detecta um pedido de ligação lança a aplicação cliente, iniciando-se assim a sessão para este novo cliente. O *servlet Router* é o gestor do servidor. Faz a gestão das ligações entre todos os clientes.

Actividades do Servlet SessionLogin

Após o pedido de ligação por parte de um novo cliente lança aplicação que permite que este novo cliente entre na sessão.

Actividades do Servlet Router

- Está permanentemente à espera de ligação, através do *ServerSocket*;
- Estabelece ligação a um dado cliente através do *socket* deste;
- A interligação entre a*pplets,* ou seja, entre clientes é gerida pelo *Router*: Este faz a gestão da comunicação entre clientes fazendo assim difusão de mensagens entre clientes;
- Cada ligação que é pedida pelo cliente é que gere os seus pedidos;
- Corta ligação a pedido do cliente, destruindo o *thread* e o s*ocket* deste.

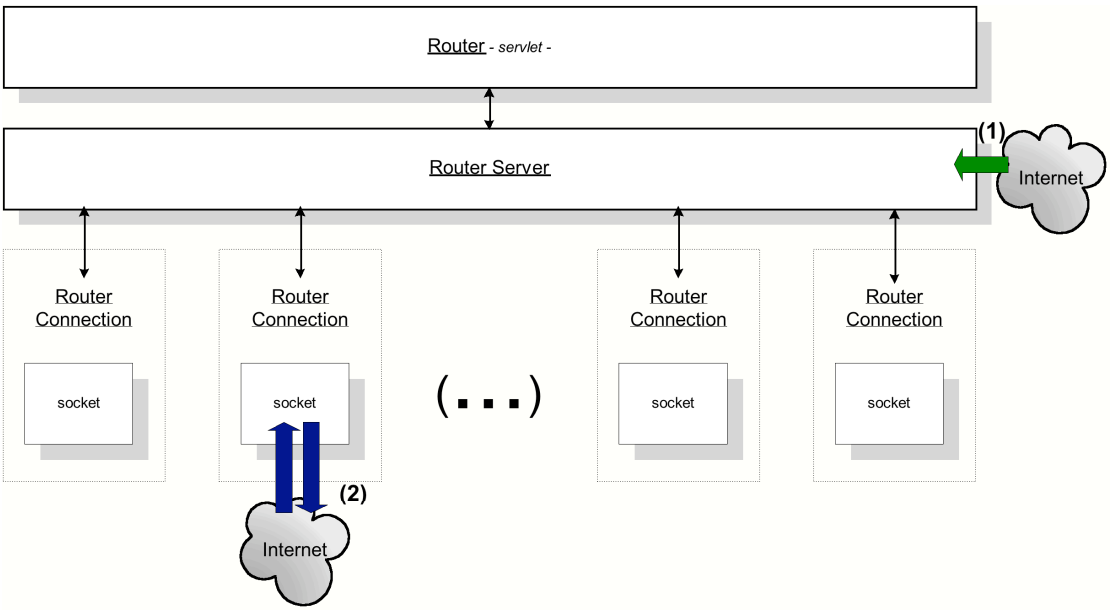

Figura 12 - Fluxo de dados entre as classes do servidor.

A seta (1) a verde, visível na **Figura 12**, representa o pedido de ligação efectuado por um novo cliente.
As setas (2) a azul, visíveis na **Figura 12**, representa o fluxo de informação entre um cliente e o servidor. A seta ascendente representa a informação que é enviada pelo cliente emissor. A seta descendente representa a informação que é recebida por todos os clientes.
As reticências indicam que pode ser referenciado tantas vezes o *RouterConnection*, quanto o número de clientes que estabelece uma ligação. O número de *sockets* criados pelo *RouterServer* é assim idêntico ao número de clientes existente na sessão.

4.3.3.1.1 Classes

As classes mais importantes para a implementação do servidor são *Router, RouterServer* e *RouterConnection*. Antes de apresentar a hierarquia das classes descrevemos cada uma das classes que constitui a parte do servidor. A explicação da funcionalidade é complementada com um diagrama de classes representado em *Unified Modelling Language* (UML).
No lado do servidor existe dois servlets, o *SessionLogin* e o *Router*.

4.3.3.1.1.1 Funcionamento do SessionLogin

O *SessionLogin* surge da necessidade de haver um *servlet* que após um pedido lance o *applet* cliente. Este *servlet* embora estando sempre activo só entra em funcionamento quando tem um pedido de entrada na sessão, referido como a seta (1) a verde na **Figura 12**.

4.3.3.1.1.2 Funcionamento do Router

Como já foi referido anteriormente, o *Router* tem a sua actividade durante a sessão.
O *Router* tem como funcionalidade responder aos pedidos de POST *http*. Os pedidos que são respondidos são os de iniciar e parar o servidor ,isto é, lançar e terminar o *thread RouterServer* e os *threads RouterConnection*. Também existe um pedido do estado do servidor. Os pedidos são atendidos e uma resposta é enviada através de uma página *html*.
Ao inicializar o *RouterServer* constrói uma lista de clientes onde armazena as ligações que vão sendo efectuadas. A partir desse momento, o *RouterServer* pode fazer a difusão dos pacotes para todos os clientes.

4.3.3.1.1.3 Funcionamento do RouterServer

O *RouterServer* é um *thread* que cria um *ServerSocket* e fica a espera de ligações dos clientes. Quando uma ligação é estabelecida o *socket* é devolvido a um *thread RouterConnection* para ser gerido pelo mesmo.
Está a seu cargo a tarefa de difundir tudo o que recebe dos clientes. Quando o *RouterConnection* detecta um pacote no *stream* de entrada então esse pacote é enviado para todos os clientes. Envia para cada elemento da lista de ligações que mantém, incluindo o próprio emissor. Para o envio, o pacote é escrito no *stream* de saída do *RouterConnection*.

4.3.3.1.1.4 Funcionamento do RouterConnection

O *RouterConnection* é um *thread* que é lançado cada vez que um cliente estabelece uma ligação com o servidor. Serve para receber as tramas do cliente e difundi-las para os outros clientes.
Embora seja o *RouterServer* que gere as ligações, é através do *RouterConnection* que o acesso ao exterior é possível visto que é a este nível que se encontram os *streams* de entrada/saída de pacotes. Resta ainda salientar que um pacote na entrada só tem um cliente emissor enquanto que um pacote na saída tem todos os clientes possíveis como receptores.

4.3.3.1.2 Esquema total da hierarquia de classes do servidor

O esquema total consiste na apresentação de um diagrama de classes usando UML, como pode ver na **Figura 13**. O diagrama de classes descreve os atributos e os métodos (o construtor, quando existe, e as outras operações) de cada classe bem como a hierarquia e relações entre classes. Para uma melhor compreensão da notação usada nos diagramas de classes UML consultar o **Anexo C** ou [11][17][18].
Note-se que a classe *ServerSocket* é criada pelo próprio Java™ podendo encontrada explicações em http://java.sun.com .

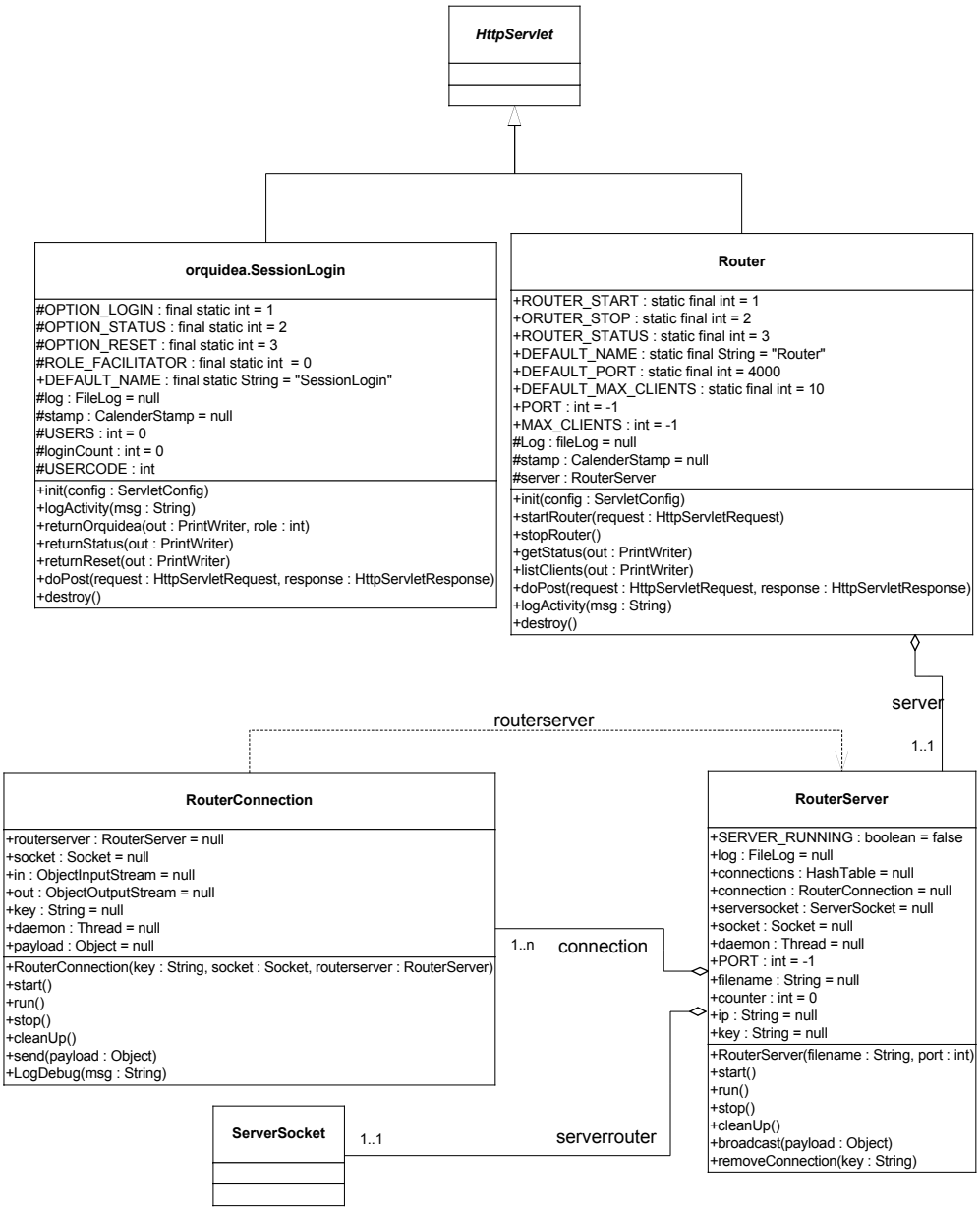

Figura 13 - Diagrama de classes do servidor

4.3.3.2 Cliente

O cliente não é mais que um *applet* que pode ser carregado através de uma página *www*.

***Actividades do Applet* Cliente**

- Permite a emissão de pedidos:
  - De ligação (início da reunião, actividade desencadeada pela submissão do *Login* ao *servlet SessionLogin*).
  - Para sair da reunião, eliminando a sua ligação.
- Enviar acções/mensagens/comentários para outros Clientes (*applets*).
- Mostrar as mensagens/comentários.

O Cliente na sua criação cria também *sockets* que usa para a ligação que estabelece com os outros clientes.
A **Figura 14** mostra a forma como a interacção de um interveniente é difundida para a aplicação dos outros intervenientes. A implementação consiste na criação de canais virtuais de comunicação permitindo que as comunicações façam-se ao nível do *SessionDialog*, *SessionChart*, P*articipants* e *SessionToken* de cada um dos clientes (representados na zona (3), a verde, na **Figura 14**).

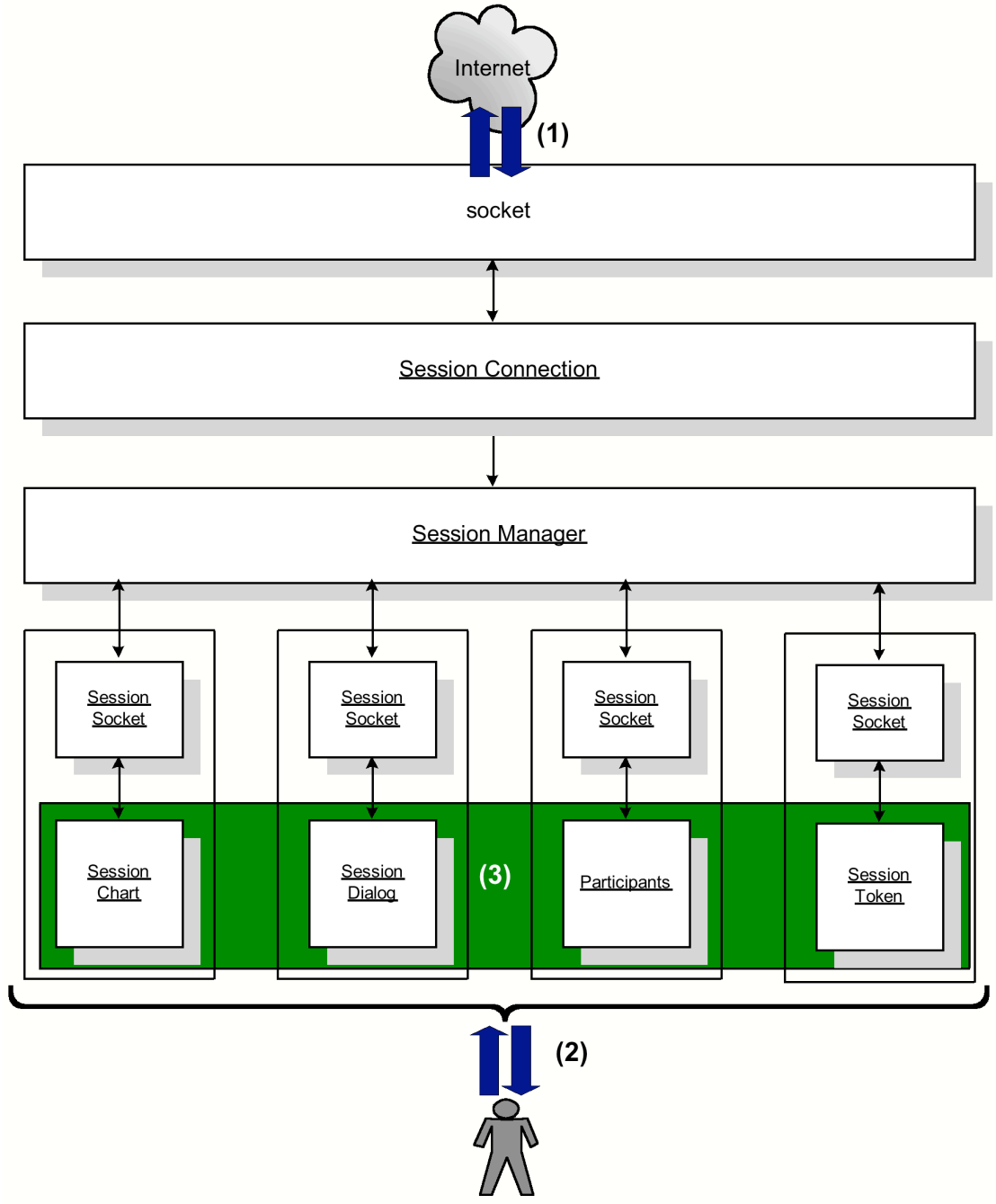

Figura 14 - Fluxo de dados entre as classes do cliente.

As setas (1) a azul na **Figura 14** representam o fluxo de informação que entra e sai da aplicação cliente. A informação que sai de um emissor deve chegar a todos os clientes.

As setas (2) a azul na **Figura 14** representam o fluxo de informação que é gerado pelo e para o interveniente. Corresponde à informação introduzida e visualizada pelo utilizador.
Das quatro áreas distintas na zona (3) a verde da **Figura 14**, o *SessionDialog* e o *SessionChart*, já abordados anteriormente, na descrição do modelo em 4.2.4.1 e 4.2.4.4 respectivamente, corresponde às áreas mais importantes de intervenção por parte do utilizador. O *SessionDialog* consiste na zona onde é implementado um *IRC* e o *SessionChart* corresponde à zona onde é implementado um *flipchart*. As outras duas áreas, embora menos importantes para o utilizador, têm um papel vital para a gestão que é feita durante toda a sessão. O *Participants* dá acesso à lista dos participantes da sessão. O *SessionToken* faz a gestão do *token*. O *token* surge da necessidade de haver exclusividade quando alguém cria ou altera informação no *SessionChart*, salvaguardando da manipulação por dois utilizadores do mesmo elemento.

4.3.3.2.1 Classes

As principais classes que constituem o cliente são *orquideaApplet*, *SessionManager*, *SessionConnection*, *SessionSocket*, *Login*, *SessionDialog, SessionChart, Participants e Token*. As últimas quatro classes já foram abordadas na secção anterior visto serem as classes com as quais o interveniente tem mais interacção durante a sessão.
Antes de apresentar a hierarquia das classes é feita uma descrição de cada uma das classes que constitui a parte do cliente. A explicação da funcionalidade é complementada com um diagrama de classes usando UML e com a descrição dos atributos e métodos de cada classe no **Anexo D**.
Note-se que o cliente não é constituído apenas por estas classes mas a descrição destas permite uma visão global sem entrar em demasiado detalhe de implementação e desfocando do objectivo que é dar a visão das classes alicerce para o cliente.

4.3.3.2.1.1 *Funcionamento do* orquideaApplet

É a aplicação cliente, ou seja, o *applet* onde a aplicação funciona e onde tudo é gerido. Quando o *applet* é lançado, são inicializadas as classes *Login*, *Participants*, *Token*, *SessionChart*, *SessionDialog*, e o carregamento do *applet* é feito para a janela destino. Enquanto activo funciona como interface para o utilizador.
Existe uma distinção entre a aplicação para o facilitador e a aplicação para o participante, que é descrita no funcionamento da aplicação na secção 5.

4.3.3.2.1.2 *Funcionamento do* SessionManager

O *SessionManager* consiste num gestor da aplicação que lança um *thread* para ler pacotes do *buffer* do *SessionConnection*. Quando existe um pacote não nulo na ligação então esse pacote é processado.
O processamento do pacote consiste em analisar o cabeçalho do pacote. Mediante o comando envia o pacote através do *socket* do *SessionConnection* para um destino específico ou armazena o pacote. A este nível é feita a gestão de *sockets* com a manutenção da lista dos *sockets* e a atribuição de portos aos *sockets,* assim como a distribuição de pacotes.

4.3.3.2.1.3 *Funcionamento do* SessionConnection

O *SessionConnection* usa um *buffer* para armazenar os pacotes em circulação. Quando é estabelecida uma nova ligação cria um novo *socket* e estabelece dois *streams*, um no sentido da entrada de informação e outro no sentido da saída da informação. Lança um *thread* para a gestão desta ligação. Enquanto houver ligação o *SessionConnection* está activo. Nessa situação, está sempre a tentar ler do *stream* de entrada do *socket* um pacote. Quando detecta coloca o pacote no *buffer*.
A destruição de uma ligação faz com que seja desactivado o processo de permanecer em escuta por novos pacotes.

4.3.3.2.1.4 *Funcionamento do* SessionSocket

Trata-se de uma classe que permite o estabelecimento de canais virtuais entre as mesmas áreas remotamente. Mais concretamente, o *SessionSocket* permite que a comunicação seja feita entre as mesmas classes. Para isso cada classe que tem uma intervenção por parte de um interveniente tem um

SessionSocket para enviar e cada um dos outros intervenientes possuem um *SessionSocket* na classe idêntica para receber. Esta classe apenas cria *sockets* e envia informação para a classe *SessionManager* informando da sua existência. O *SessionManager* apenas adiciona o *socket* à lista de *sockets*.

4.3.3.2.1.5 *Funcionamento do* Login

A classe *Login* é activada através da resposta ao POST feito ao *servlet SessionLogin*. Esta classe tem o seu desempenho quando o interveniente entra na sessão. Corresponde à interface com a qual o interveniente interage para introduzir os seus dados pessoais, antes de entrar na sessão. Pode eventualmente ocorrer durante a sessão, mas sempre quando o interveniente sai da sessão e volta a entrar. Quando todos os dados estão introduzidos esta classe deixa de ter interface visível e tornando visível a interface com que o utilizador interage durante a sessão.

4.3.3.2.1.6 Funcionamento do SessionDialog

O *SessionDialog* é constituído por duas zonas, uma para introdução de mensagens e outra para a visualização destas. Tem ainda o botão *Send* para que as mensagens sejam enviadas após o seu accionamento.
O fluxo de mensagens no *SessionDialog* inicia-se com o envio, após a criação e introdução de informação. A mensagem surge então no interveniente receptor.

4.3.3.2.1.7 Funcionamento do SessionChart

O *SessionChart* é constituído por três áreas. Uma contém a zona de desenho, outra, relacionada directamente com a área de desenho, que permite manipular e ter informação da área de desenho e a última que contém o conjunto de elementos gráficos que representam os objectos que o interveniente pode criar.
Para a utilização do *SessionChart* é necessário ter o *token* activado. Caso não esteja activado o interveniente apenas pode limitar-se a ver o que está a ser executado por quem o tem activado.
O *token* é o mecanismo que dá permissão ao interveniente de interagir com o *SessionChart* através da criação e manipulação da informação existente.

4.3.3.2.1.8 Funcionamento de Participants

Trata-se da parte que faz a gestão da lista participantes, introduzindo um novo, removendo quem sai, quer voluntariamente ou involuntariamente, ou até permitindo o acesso aos dados introduzidos por cada interveniente aquando da sua entrada na sessão. A introdução de um novo participante consiste na introdução do *nickname* desse interveniente na lista de participantes assim como todos os dados que este pode preencher quando inicia a sessão. Apenas o preenchimento do *nickname* é obrigatório.
A remoção da lista dá-se de forma voluntária quando o interveniente toma a decisão de sair fechando a janela da aplicação onde decorre a sessão. A remoção involuntária ocorre quando uma ligação não é bem estabelecida e, por detecção do servidor, a ligação é deitada a baixo e o interveniente tem de voltar a entrar na sessão.
A classe *Participants* tem um algoritmo inerente ao seu funcionamento. Consiste no envio de pacotes específicos de entrada e saída de participantes da sessão. É estabelecido o protocolo de *shaking hands*. Quando um novo cliente estabelece a ligação, este envia um pacote informando os restantes da sua existência, após a recepção, os restantes enviam de retorno um pacote com a sua informação e informação da reunião. Com a informação de cada cliente, o novo cliente cria a sua lista de participantes. Com a informação sobre a reunião o novo cliente pode actualizar a informação tendo acesso a tudo o que foi gerado até então.
A lista de participantes tem uma organização FIFO (*First In First Out*), ou seja o primeiro a entrar é o último da lista. Por omissão, o facilitador, independentemente da ordem de chegada, é introduzido na lista como sendo a primeira entrada, ficando por isso no fim da lista, como pode verificar na secção 5, na **Figura 28**.
Quando um cliente sai da sessão, terminando a ligação, este envia um pacote a avisar os restantes de tal acção, assim estes actualizam a sua lista de participantes removendo-o dela.

4.3.3.2.1.9 *Funcionamento do* SessionToken

O conceito de *token* é aqui introduzido pois há a necessidade de garantir exclusividade na edição e/ou alteração da área de desenho *SessionChart*.
A gestão do *token* consiste na atribuição do *token* a um interveniente. A activação do *token* é inicializada pelo facilitador. Quando este desactiva o *token*, o *SessionToken* acede à lista de participantes e atribui ao próximo elemento da lista de participantes, caso esta só tenha o facilitador volta a activar o *token* do facilitador.
O algoritmo inerente ao funcionamento do *SessionToken* consiste na atribuição do *token* segundo a ordem da lista. A atribuição do *token* inicia-se com o envio de um pacote ao primeiro participante que entrou. Por omissão, esse é sempre o facilitador e é este que tem a função de iniciar o ciclo. Após o facilitador seguem-se os restantes participantes que recebem igualmente um pacote indicando a posse do *token*. Qualquer participante que receba o *token* tem a autorização para editar o *SessionChart*. Quando termina as suas funções nesta área envia um pacote com o *token* para o próximo participante e assim sucessivamente.
Este ciclo pode ser quebrado com a requisição do *token* por parte do facilitador que retoma o *token* e volta a iniciar o ciclo. O processo efectua-se da mesma forma que a circulação do *token* mas neste caso o pacote enviado tem como função retirar o *token* ao participante que o detém e retornar ao facilitador.
É enviada a informação a todos os participantes de que o facilitador detêm o *token*.

4.3.3.2.2 Esquema total da hierarquia de classes do cliente

O esquema total consiste na apresentação de um diagrama de classes usando UML, como pode ver na **Figura 15** O diagrama de classes descreve apenas a hierarquia e relações entre classes. Para uma melhor compreensão da notação usada nos diagramas de classes UML consultar o **Anexo C** ou [11][17][18].

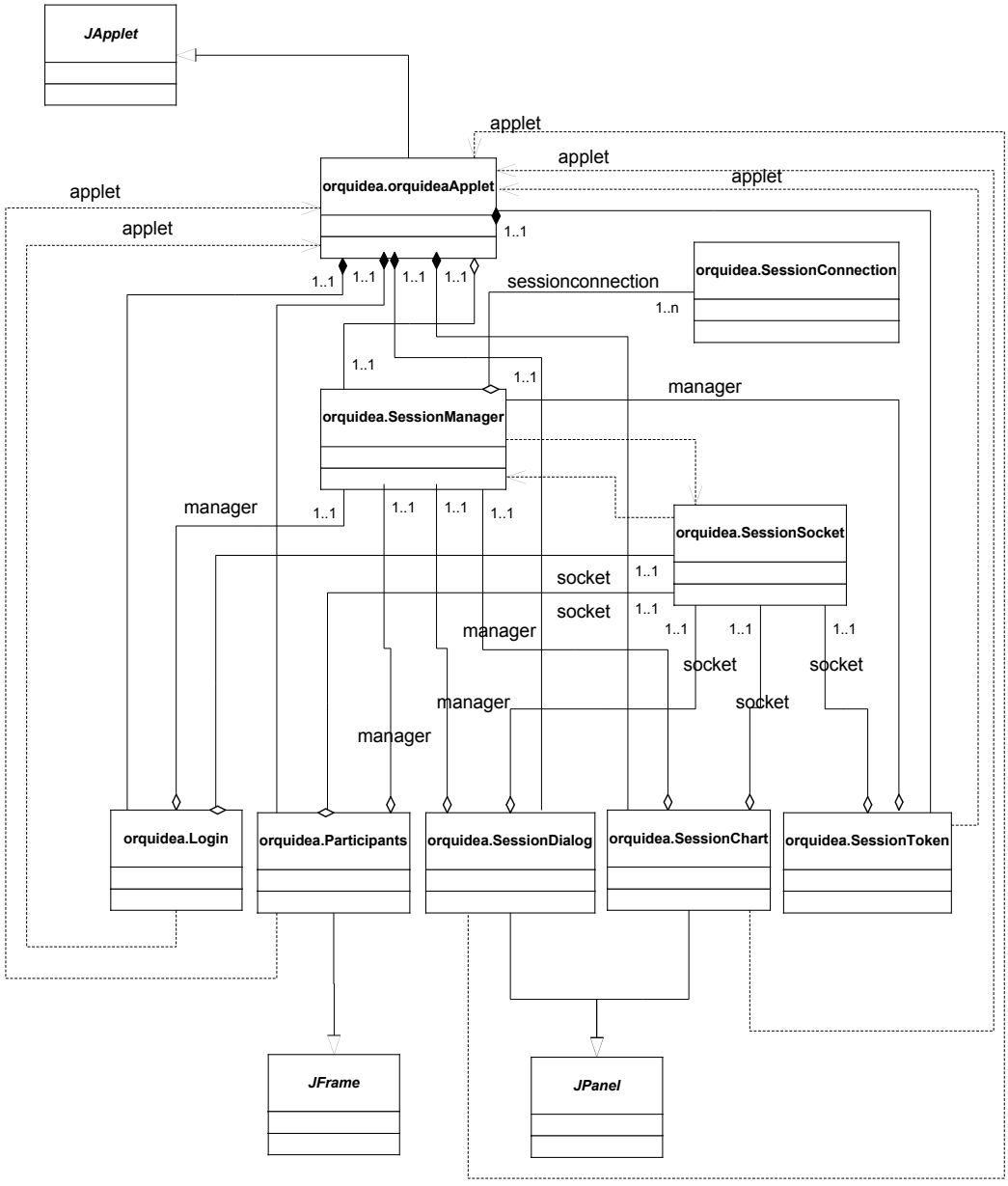

Figura 15 - Diagrama das classes do cliente

Note-se que no diagrama da **Figura 15** não foi incluído a descrição do construtor e métodos das classes pela razão de simplificar a representação. A representação em notação UML, de cada classe que constitui o diagrama de classes da **Figura 15**, com a descrição dos atributos e métodos, encontra-se no **Anexo D**.

4.3.3.3 Técnicas usadas para a resolução de problemas detectados durante a implementação

Os aspectos referidos nesta secção são aspectos de implementação que foram decididos tendo um fundamento explicativo para tal. Assim,

- Para a recepção de informação entre clientes surge a vontade de estabelecer um protocolo de comunicação que permite a comunicação entre classes idênticas mas em *applets* diferentes. Para isso, qualquer pacote usado na comunicação tem um formato pré-definido. O pacote é constituído por um cabeçalho e por um corpo. O cabeçalho contém sempre um comando e o cliente receptor quando recebe um pacote lê esse o cabeçalho e encaminha para o destino certo.
- Na entrada de um novo cliente é necessário que este saiba quem já está na sessão e vice versa. Assim surge a necessidade de implementar o protocolo de *Shaking hands*, descrito no funcionamento da classe *Participants*, na secção 4.3.3.2.1.8. Este protocolo é ainda usado para o caso de um interveniente sair prematuramente da sessão.
- Na interacção com o *SessionChart*, por ser uma área comum de interacção, surge a necessidade de garantir que esta área só é usada por um interveniente de cada vez. Assim é introduzido o conceito de *token*, muito usado em Redes de Telecomunicações e que consiste na passagem de testemunho de mão em mão. Este conceito já foi descrito no funcionamento do *SessionToken*, na secção 4.3.3.2.1.9.
- O carregamento da aplicação cliente não é feita de uma forma muito rápida e para resolver essa situação introduz-se informação no *status bar* da evolução do carregamento.

4.3.3.4 Opções de implementação

Consiste na descrição pormenorizada das opções tomadas tendo em conta a funcionalidade e utilização tida como ideal para a aplicação assim como as restrições técnicas quer no processo de implementação quer na utilização do meio de comunicação, a *Internet*.

4.3.3.4.1 Funcionamento da aplicação

- O acesso aos dados da Agenda não é muito desenvolvido pois sai fora do âmbito deste projecto.
- O bloco de folhas é constituído inicialmente por duas páginas com a primeira para a lista de tópicos da Agenda e a segunda para a introdução de informação.
- Restrição do campo de acção do *SessionChart* às acções mais comuns. O utilizador apenas pretende um meio de expressão rápido e eficaz em reuniões remotas. Também é tido em conta o tipo de utilizador típico, que ao reduzir as opções de interacção atingirá a confiança na interacção mais rapidamente tendo menor possibilidade de desistir nos primeiros momentos de interacção.
- Redução da liberdade de acção do utilizador na criação de elementos gráficos no *SessionChart*. Ao desenvolvermos essa área estaríamos a desviar do verdadeiro objectivo deste projecto. No entanto não deixamos de achar positiva a ideia de permitir uma maior liberdade ao utilizador neste campo.
- Num situação em que a sessão já for iniciada e já haja um facilitador, o novo cliente com o papel de facilitador irá substituir o antigo facilitador. Para garantir que isso não acontece é necessário definir os papéis previamente. Actividade essa que deve estar a cargo do facilitador na fase de pré-reunião.
- Quando um interveniente não tem o *token* activado é possível a visualização, no seu *SessionChart*, de todos as acções por parte de quem tem o *token* activado. Quem não tem o *token* activado fica com a folha fixa, podendo apenas por intervenção de quem tem *token*, com a criação de uma nova folha.

4.3.3.4.2 Restrições técnicas na Implementação

- O *applet* só pode responder ao servidor que o lançou;
- Impossibilidade de arquivar localmente a informação gerada. Esta tem sempre de ser guardada pelo servidor.

A última restrição impediu a implementação de um mecanismo para gravar o contexto permitindo um acesso posterior.

Para uma análise em maior detalhe enconta o código gerado para a implementação das classes descritas nas secções anteriores no **Anexo B**.

5 RESULTADOS

5.1 FUNCIONAMENTO DO NETCHART

O resultado consiste numa aplicação, chamada NETCHART que pode ser carregada através da *Internet*, através de uma página *html*, e com a qual pode-se organizar uma reunião remota. Após a prévia instalação, descrita no **Anexo A**, do *Plug-In* que acompanha o *Java™ 2 Runtime Environment,* inicia o processo de carregamento da aplicação através do elo ***Login*** da página, visível na **Figura 16**.

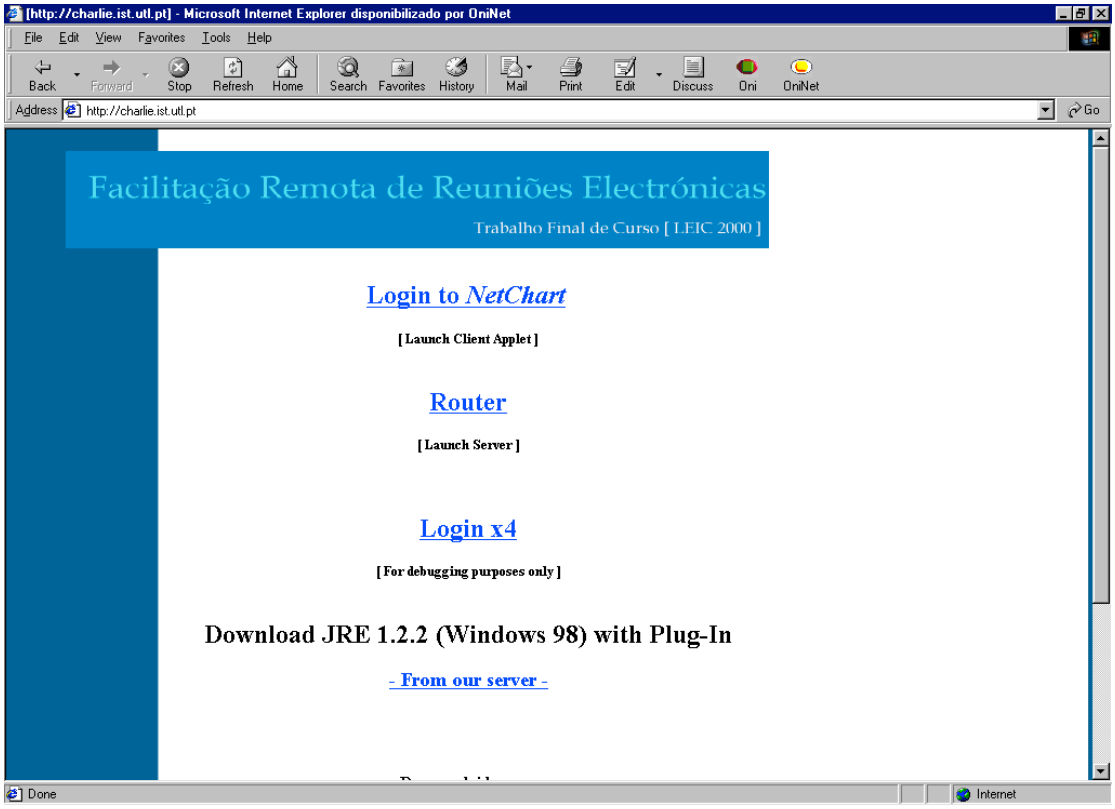

Figura 16 - Página principal para o carregamento do NETCHART.

Tem escolher o papel a desempenhar, como pode ver na **Figura 17**, e mediante isso é carregada a interface para introdução dos dados pessoais, como o papel a desempenhar correspondente, como pode verificar na **Figura 18** e na **Figura 19**.

Figura 17 - Página para escolha do papel a desempenhar na sessão.

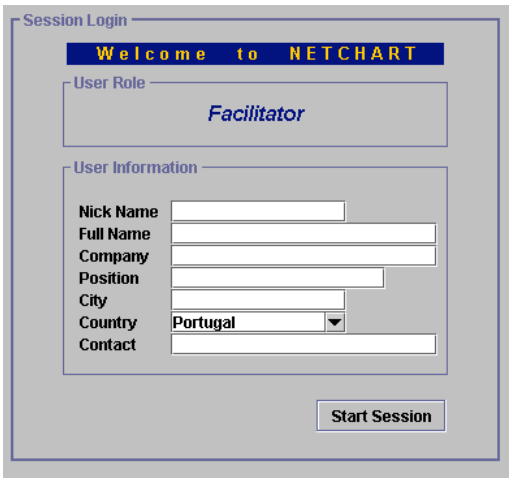

Figura 18 - Introdução de dados do facilitador.

Figura 19 - Introdução de dados do participante.

A introdução do campo *nickname* é obrigatória, sendo o preenchimento dos restantes campos facultativo. O campo *nickname* pode conter um nome fictício, permitindo o anonimato do interveniente. O campo *Country* é o único previamente preenchido e se não for alterado introduz o País visível nos dados pessoais do interveniente.
Só deve existir um facilitador. Se alguém aceder como facilitador, já existindo um na sessão, faz com que a gestão do *token* passe a não funcionar correctamente. A razão para tal é que o *token* é um mecanismo que dá acesso a uma zona comum e se houver dois a concorrer a gestão não sabe a quem deve atribuir o *token*. Assim o facilitador tem como tarefa definir previamente os papéis informando todos antes do início da sessão para evitar a situação descrita.

Após o accionamento do botão *Start Session*, visível nas **Figura 18** e **Figura 19**, torna-se visível a interface com que o interveniente vai interagir até ao fim da sessão, como é um aplicação que pode ver na **Figura 20**. O carregamento da aplicação cliente não é feita de uma forma muito rápida mas o utilizador tem a percepção de tudo o que está a acontecer, através do *status bar* da aplicação.

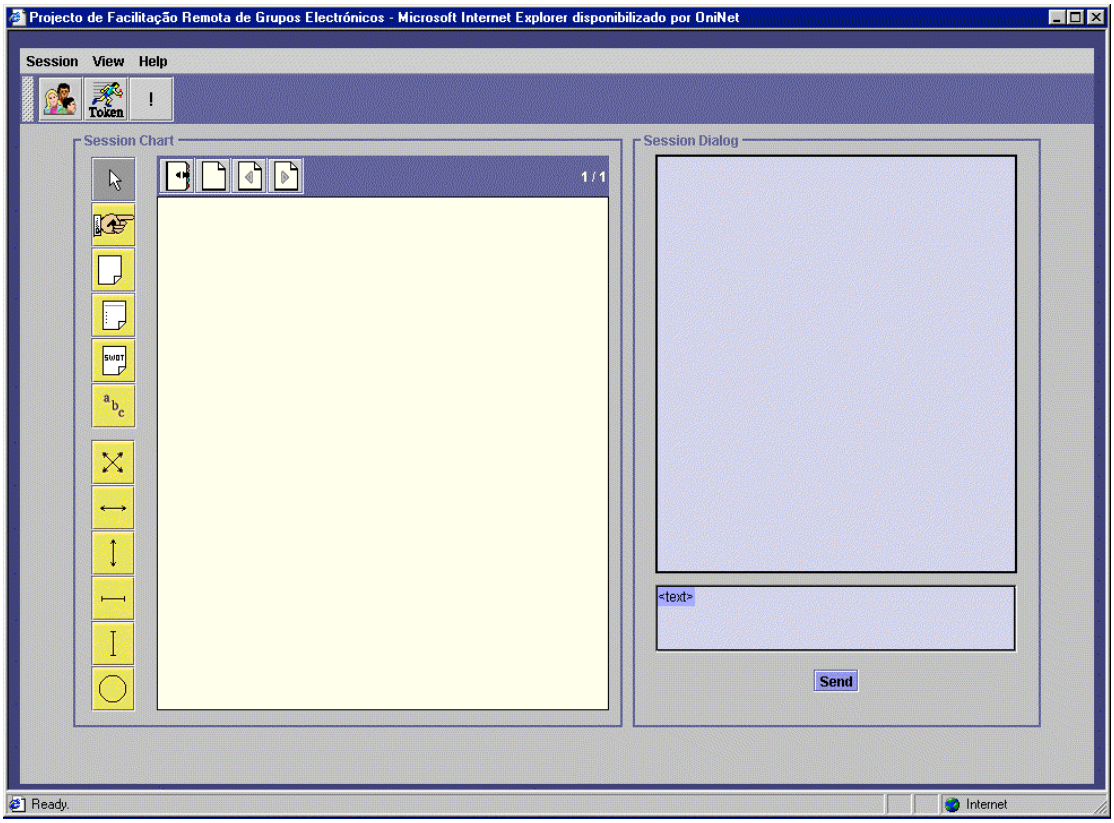

Figura 20 - Aspecto geral do NETCHART.

É visível a distinção entre as duas principais áreas de intervenção, o *SessionChart* e o *SessionDialog*. Existe um menu que permite as operações de:

- Sair da Sessão;
- Visualizar apenas o *SessionChart* ou o *SessionDialog* ou ambos e;
- Dar uma ajuda na forma como funciona a aplicação.

Estas operações são visíveis nas **Figura 21**, **Figura 22** e **Figura 23**.

Figura 21 - Operação para saída da sessão através de *Session*.

Figura 22 - Operações de visualização através de *View*.

Figura 23 - Operação de ajuda através de *Help*.

Existe também um *tollbar* no canto superior esquerdo da aplicação. Este permite o controlo e acesso a informação da sessão. Como pode ver nas **Figura 24** e **Figura 25**, existe uma distinção entra a aplicação facilitador da aplicação participante.

Figura 24 - *Toolbar* da aplicação facilitador.

Figura 25 - *Toolbar* da aplicação participante.

O botão extra existente no *toolbar* da aplicação facilitador, visível na **Figura 24**, permite que este recupere o *Token* e que quando activado na sua aplicação possa iniciar o ciclo de atribuição aos restantes participantes. O facilitador tem assim a tarefa de iniciar este ciclo com a activação do botão direito. O botão **Token** é automaticamente activado, como é visível na **Figura 26**.

Figura 26 - Botão de emergência para a recuperação do *Token*.

O botão à esquerda, quando premido, como é visível na **Figura 27**, permite o aceso à lista de participantes, como pode ver na **Figura 28**, podendo consultar os dados de cada interveniente premindo cada nome da lista de *nickmanes*.

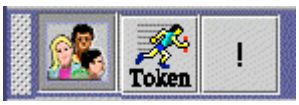

Figura 27 - Botão dos Participantes activado.

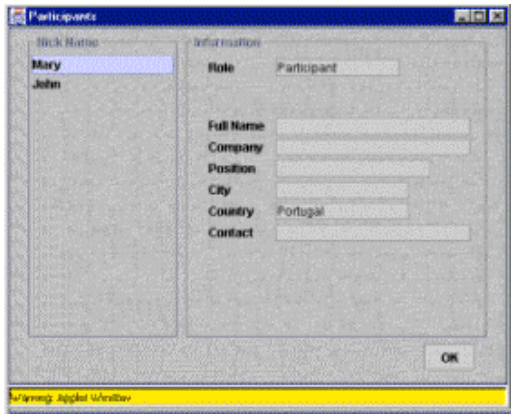

Figura 28 - Janela com os dados pessoais de todos os participantes.

O *SessionDialog* consiste na implementação de um *Internet Relay Chat* (IRC), onde o interveniente pode dialogar textualmente com os restantes intervenientes, como pode ver na **Figura 29**. A conversa é visível para todos.

Figura 29 - Aspecto do *SessionDialog*.

O *SessionChart* consiste na implementação de um *flipchart* electrónico onde é possível construir informação textual aliada à informação visual inerente à organização. Essa organização é visível através do estabelecimento de relações entre elementos e pela animação efectuada quando um elemento está a ser alterado, chamando a atenção dos intervenientes. Como exemplo pode ver as **Figura 30** e **Figura 31**.

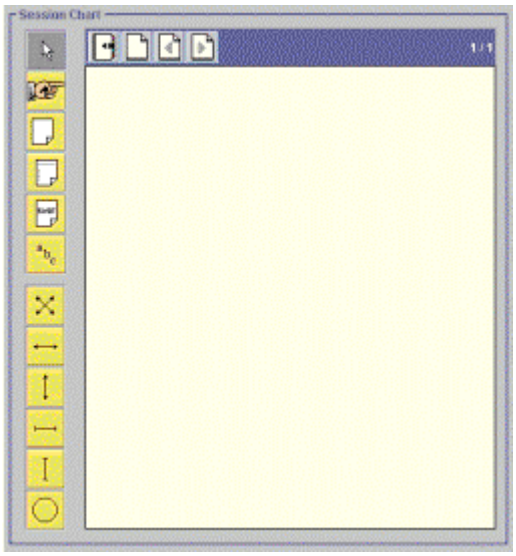

Figura 30 - Aspecto do *SessionChart*.

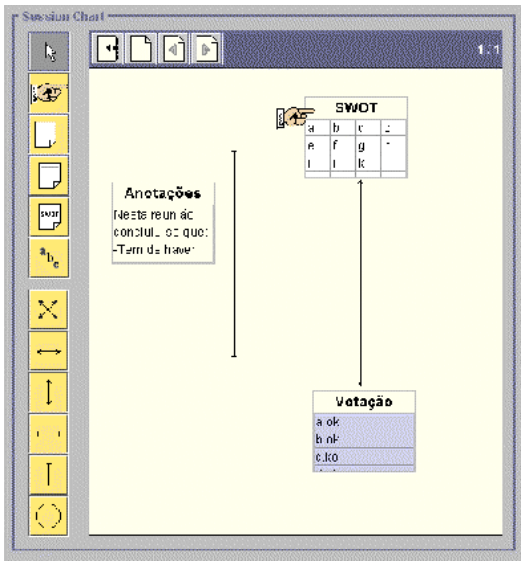

Figura 31 - Relações entre elementos.

O *SessionChart* tem um *toolbar* sobre a área de desenho, visível na **Figura 32**.

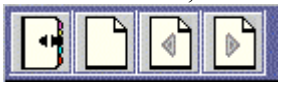

Figura 32 - *Toolbar* de controlo da área de desenho do *SessionChart*.

Seguindo a ordem de botões da esquerda para a direita, permitem ,respectivamente as operações de:

- Aceder aos tópicos da agenda;
- Criar uma nova folha de desenho;
- Retroceder nas folhas criadas e;
- Avançar nas folhas criadas.

Para o caso específico do acesso aos tópicos da agenda o que encontra é uma página pré-definida para esse efeito onde cabe ao facilitador a introdução dos dados nesta área.
Para que a edição e/ou alteração de elementos gráficos seja possível é necessário ter o botão *Token* activado, como já foi referido anteriormente, e tem de activar um elemento dos *toolbars* de elementos, visíveis na **Figura 33** e **Figura 34**, junto à área de desenho do *SessionChart*.

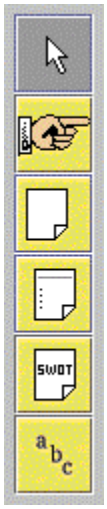

Figura 33 - *Toolbar* dos elementos textuais.

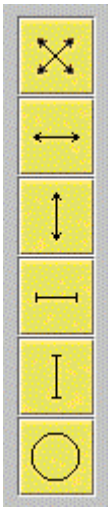

Figura 34 - *Toolbar* dos elementos gráficos.

É feita a distinção entre os elementos pois tem características diferentes, caracterizadas pelas operações possíveis para cada tipo de elemento, como pode ver nas **Figura 35**, **Figura 36** e **Figura 37**.

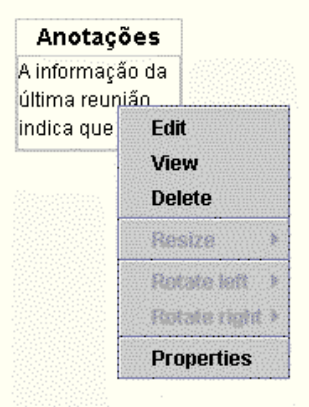

Figura 35 - Operações possíveis nos elementos textuais.

Figura 36 - Operações possíveis nos elementos gráficos tipo seta e separador.

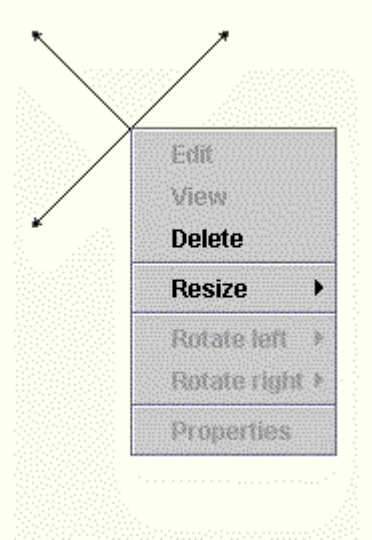

Figura 37 - Operações possíveis nos elementos gráficos do tipo setas cruzadas e círculo.

No caso específico das operações *Edit* e *View*, estas também são possíveis por atalhos com um *click* duplo sobre um elemento textual. A operação *Edit* consiste na edição dos componentes textuais do elemento, como é apresentado no exemplo das **Figura 38** e **Figura 39**, enquanto que a operação *View* permite apenas o acesso à informação sem permitir a alteração. A operação *Edit* só é permitida ao interveniente que tem o *token* activado enquanto os restantes limitam-se a poder usar *View*. Assim a operação *View* é a única que pode ser acedida por parte dos intervenientes que não têm o *token* activado.
A operação *Resize* é permitida a qualquer elemento gráfico enquanto que *Rotate Left* e *Rotate Right* só o são para as setas. Isso acontece pois os outros elementos não têm qualquer necessidade de efectuar rotações na sua disposição.

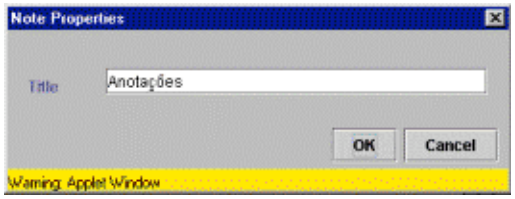

Figura 38 - Edição da componente título elemento do tipo bloco de notas.

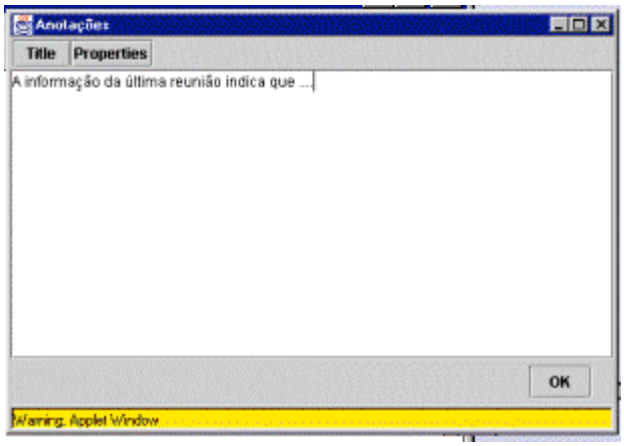

Figura 39 - Introdução de texto no bloco de notas.

É permitido ainda a alteração de posição de um elemento. A acção só é permitida ao interveniente que tem o *token* activado e é feita através da acção *drag-and-drop*, representado nas **Figura 40** e **Figura 41**.

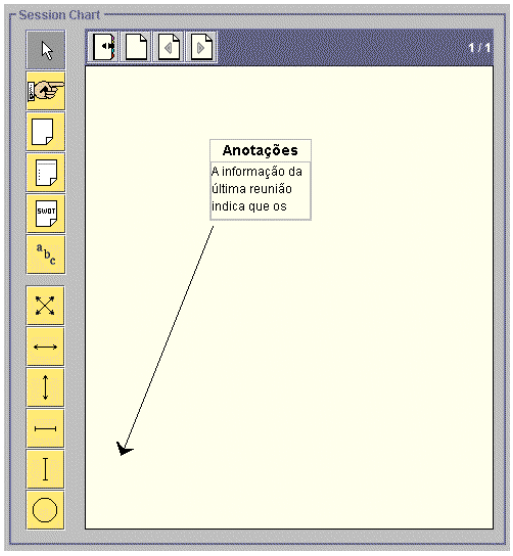

Figura 40 - Trajectória do movimento efectuado pelo rato.

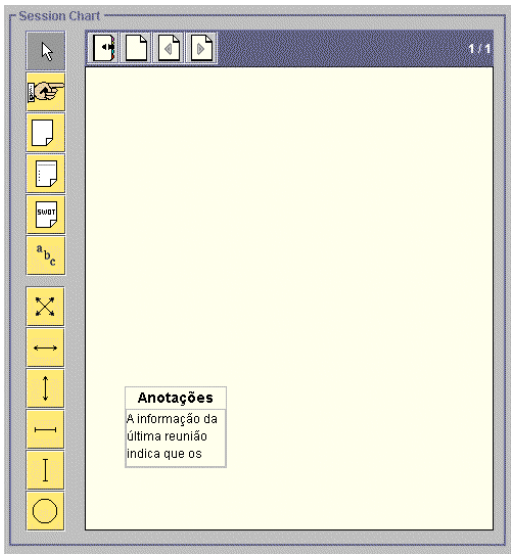

Figura 41 - Posição final do elemento

Para remover um elemento, o interveniente tem que ter o *token* activado. A operação é efectuada com a selecção da opção *Delete* visível , por exemplo, na **Figura 36**, que apaga o elemento da área de desenho, como pode ver nas **Figura 42** e **Figura 43**.

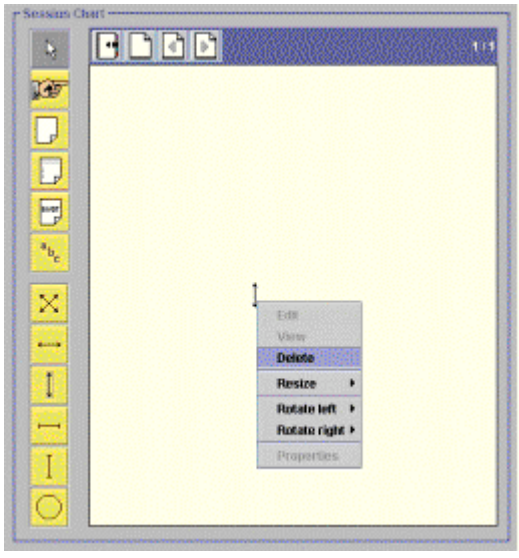

Figura 42 - Selecção da opção *Delete*.

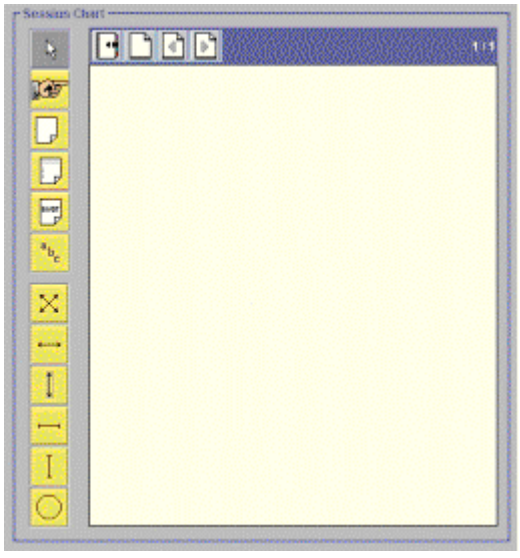

Figura 43 - Área de desenho após remoção do elemento.

Pode-se ainda copiar texto do *SessionDialog* para o *SessionChart* através da selecção do texto seguido do comando *Copy* para um elemento do tipo bloco de notas e fazendo *Paste*, como pode ver na **Figura 44** e na **Figura 45**. A transição apenas pode ser feita para um bloco de notas aberto.

Figura 44 - Cópia do texto seleccionado no *SessionDialog*.

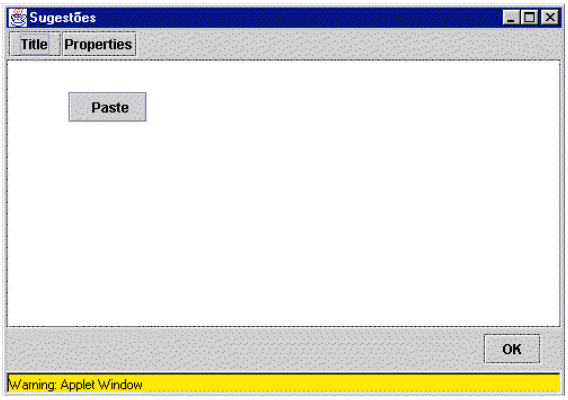

Figura 45 - Bloco de notas para arquivo da cópia.

Quando um utilizador estabelece uma ligação atrasada, este recebe automaticamente toda a informação que foi gerada até então pelos restantes utilizadores.
Para sair da sessão basta seleccionar no menu *Session* a acção *Leave*, como pode ver na **Figura 46**, e surge uma mensagem de despedida, visível na **Figura 47**.

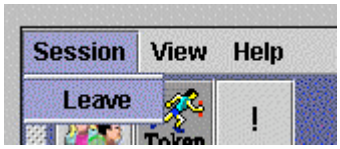

Figura 46 - Selecção da opção para sair da sessão.

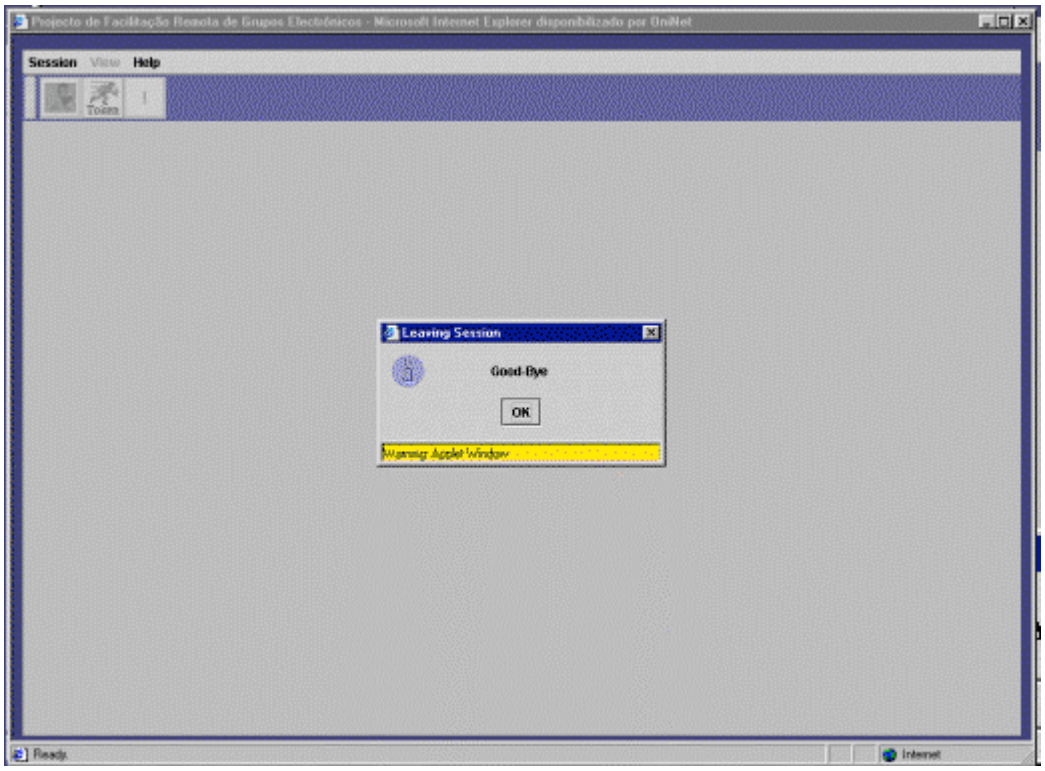

Figura 47 - Mensagem de despedida.

Após fechar a caixa com a mensagem de despedida, visível na figura anterior, tem a possibilidade de entrar novamente na sessão através do elo ***To login click here***, como pode ver na **Figura 48**.

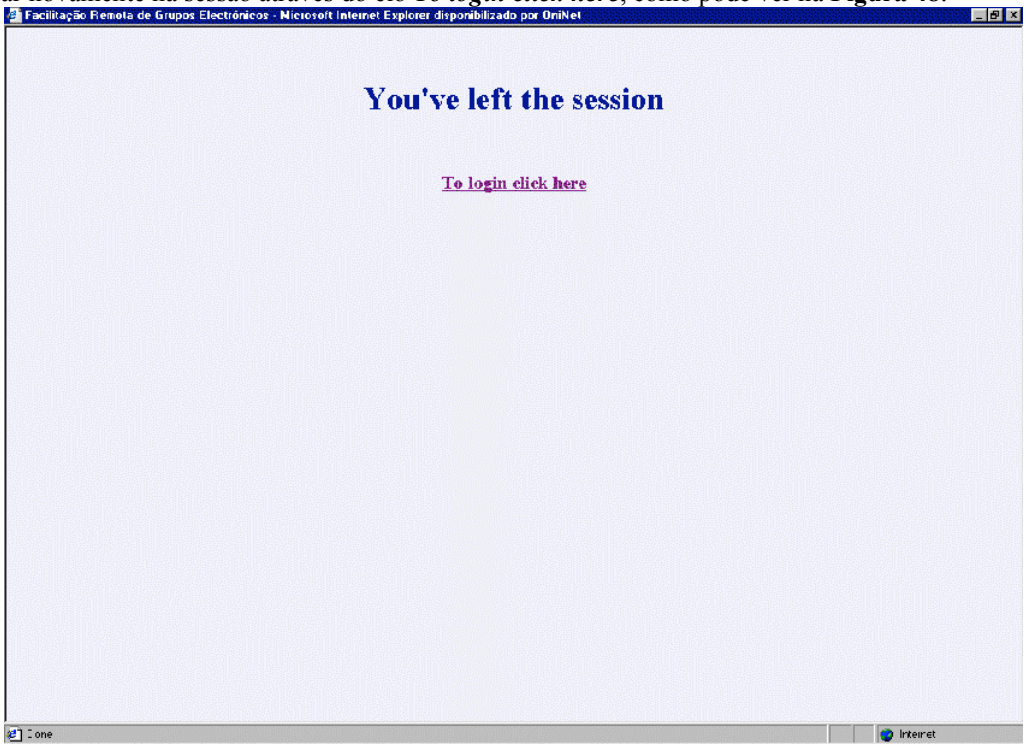

Figura 48 - Elo para a entrada na sessão após a saída.

5.2 REGRAS PARA O CORRECTO FUNCIONAMENTO DO NETCHART

- Antes da reunião o facilitador tem como tarefa definir o papel de cada interveniente;
- O facilitador deve tentar não se atrasar na entrada da sessão. Isto porque é o único interveniente que pode iniciar a circulação do *token*. Sem facilitador nada pode ser feito no *SessionChart* do NETCHART;
- No caso em que uma ligação não é bem estabelecida, detectada ou através da lista de participantes vazia ou por não receber nada do que está a ser criado, o interveniente deve sair da sessão e voltar a entrar.
- O utilizador não pode aceder à reunião através de um computador ligado a uma rede local que tenha barreiras de segurança elevadas, do tipo *firewalls*, que não permitem o carregamento da aplicação por uma entidade externa à rede local.

Pode encontrar a descrição promenorizada da aplicação NETCHART no **Anexo A**.

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

A aplicação resultante resume-se na implementação de um *flipchar*t electrónico para reuniões remotas que é complementado por um IRC.
A aplicação permite a exposição de ideias de uma forma muito semelhante a usada numa reunião convencional, onde seja usado um *flipchart*. Esta cria uma linguagem gráfica que leva a uma maior criatividade e expressividade de ideias. Permite dizer mais em menos tempo pois para além da informação textual permite estabelecer relações e distinções entre aspectos abordados.
A implementação de um modelo de funcionamento próximo do modelo mental para a utilização do *flipchart* convencional reduz a fase de adaptação à aplicação. A sua implementação electronicamente permite a exploração das vantagens em usar uma tecnologia diferente, como mecanismo para criar e alterar de uma forma rápida, copiar informação entre outros aspectos.
Para além do tempo poupado na utilização de meios que permitem uma correcção mais rápida o tempo dispendido para a realização da reunião é reduzido em relação a uma reunião convencional.
A utilização da *Internet* como meio de comunicação permite grande acessibilidade da aplicação. Assim, devido à globalização deste meio existe menos restrições na realização de uma reunião remota.
A utilização de tecnologias muito actualizadas na criação da aplicação cliente-servidor e a utilização de uma linguagem que é independente da plataforma permitiu a obtenção dos objectivos propostos.
Um aspecto ao qual foi dado especial relevância foi a modularidade do código gerado que permite a integração de ou em outros módulos tudo o que foi desenvolvido neste projecto.

Como perspectivas futuras no que diz respeito a novos desenvolvimentos para o melhoramento da solução apresentada destaca-se aspectos para a parte cliente e aspectos para a aplicação servidor:

Cliente

- Criar a possibilidade de arquivar a informação gerada durante a reunião;
- Desenvolver a componente gráfica da aplicação com a introdução de mais operações e variantes das existentes , permitindo uma maior liberdade de criação e alteração;
- Introduzir na aplicação facilitador uma componente estatística da sessão dando uma ideia da participação de cada interveniente. O facilitador pode assim incentivar ou dar o seu suporte as menos participativos;
- Introduzir uma lista de frases mais usadas num ambiente de reunião ou criar componentes gráficas que transmitam a mesma ideia;
- Introduzir as componentes áudio e vídeo.

Servidor

- Delegar no servidor a tarefa de guardar o contexto e informação de uma sessão permitindo a recuperação posterior;
- Passar a responsabilidade de manter a lista de clientes ligados e ser este o responsável pela difusão desta informação aos novos clientes;
- A gestão do *token* pode ser igualmente transferida para o servidor.

7 BIBLIOGRAFIA

[1] Antunes, P., Guimarães, N., 1996. "User-Interface Support to Group Interaction".

[2] Antunes, P., Ho, T., 1999. "Facilitation Tool - A tool to assist facilitators managing Group Decision Support Systems".

[3] Antunes, P., Ho, T., 1999. " The design of a GDSS meeting preparation tool".

[4] Bostrom, Robert P., Anson, Robert, Clawson, Vikki K., Cap 8 "Group Facilitation and Group Support Systems".

[5] Costa, Carlos J., Ho, T., Antunes, P., 1999. "Facilitating Organizational Activities using Plans and Audits".

[6] Executive Digest, Outubro 1998. "50 melhores citações de sempre".

[7] "GDSS: Limitações e Oportunidades ".

[8] Ho, T., 1997. "Relatório de Actividades - Decisão e Criatividade em Grupos Electrónicos e Virtuais".

[9] Ho, T., Antunes, P., 1999. "Developing a Tool to Assist Electronic Facilitation of Decision-Making Groups".

[10] Ho, T., 1999. " Ferramentas de Suporte à Facilitação em Processos de Decisão em Grupo".

[11] Rumbaugh, J., Jacobson, I., Booch, G., 1999, "The Unified Modelling Language Reference Manual", Addison-Wesley ;

[12] http://java.sun.com/docs/books/tutorial/applet/appletonly/iac.html, "Sending Messages to Other Applets";

[13] http://java.sun.com/docs/books/tutorial/servlets/overview/index.html, "Lesson: Overview of Servlets";

[14] http://midir.ucd.ie/~mbsft-4b/global/env.html ,"The GDSS environment, the technocratic view";

[15] www.grove.com/services/process_consulting1.html ;

[16] www.macrovu.com/VTVCInterEffectiveness.html ;

[17] www.rational.com/uml/, 1999, "OGM Unified Modeling Language Specification" Rational Software Corporation;

[18] www.rational.com/uml/, 1997, "UML Notation Guide", Rational Software Corporation;

8 LISTA DE FIGURAS