Pós-Graduação em Ciência da Computação

“DESIGN DE COMPONENTES EDUCACIONAIS SÍNCRONOS”

Por

ENOQUE CALVINO MELO ALVES

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, AGOSTO/2005

Universidade Federal de Pernambuco
Centro de Informática
Pós-Graduação em Ciência da Computação

ENOQUE CALVINO MELO ALVES

"DESIGN DE COMPONENTES EDUCACIONAIS SÍNCRONOS"

Este trabalho foi apresentado à pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de mestre em Ciência da Computação.

ORIENTADOR: PROF. DR. ALEX SANDRO GOMES

RECIFE, AGOSTO/2005

Alves, Enoque Calvino Melo
Design de componentes educacionais síncronos / Enoque Calvino Melo Alves. – Recife : O Autor, 2005.
111 folhas : il., tab., fig.

Dissertação (mestrado) – Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2005.

Inclui bibliografia.

1. Ciência da computação – Interface homem-máquina. 2. Aprendizagem colaborativa apoiada por computador – Groupware – Requisitos de grupo. 3. Análise de requisitos – Análise de competidores, cenários e prototipação. I. Título.

004.5	**CDU (2.ed.)**	**UFPE**
004.6	**CDD (22.ed.)**	**BC2005-592**

DEDICATÓRIA

Aos meus pais.

AGRADECIMENTOS

É um momento de grande alegria para mim a conclusão deste trabalho e é com esta mesma alegria e satisfação que aproveito para agradecer as pessoas que contribuíram para este grande momento. Sem essas pessoas jamais teria chegado até aqui. Alguns contribuíram de forma mais direta para este trabalho, enquanto outros foram importantes para que eu tenha chegado até aqui.

Ao meu orientador, Professor Alex Sandro Gomes, pelo apoio na definição do trabalho, orientação e estímulo na minha formação como pesquisador. A quem hoje eu considero como um amigo e parceiro na minha vida acadêmica.

Ao meu pai, José Alves Sobrinho, que sentou em uma carteira escolar pela primeira vez na vida aos 22 anos de idade. Homem de firme propósito e fiel aos seus ideais, um grande educador, que abdicou de muitos de seus sonhos para dar aos filhos uma educação de qualidade.

A minha mãe, Odete Melo Alves, pelas noites e madrugadas intermináveis em que ficava acordada me ensinando a matemática, que mesmo sob meus protestos e choros, ensinou-me o valor da perseverança.

A Vânia Alves, minha esposa, por suportar o meu primeiro ano de mestrado ausente, e nos anos seguintes, por se tornar minha colaboradora vindo para o mestrado, escolhendo um tema relacionado com o meu e tornando esta longa trajetória muito mais suave.

Ao meu colega e amigo Maurício Braga, por desenvolver a ferramenta Gérard. A minha grande amiga Taciana Falcão pela sua colaboração e incentivo na elaboração do primeiro cenário.

SUMÁRIO

1. INTRODUÇÃO 12

2. AMBIENTES EDUCACIONAIS SÍNCRONOS 14

2.1. APRENDIZADO COLABORATIVO APOIADO POR COMPUTADOR 14

2.1.1. Taxonomia da colaboração apoiada por computador 15

2.1.2. Groupware síncrono de aprendizagem 18

2.2. PROBLEMA 20

3. ARQUITETURAS DE *GROUPWARE* 21

3.1. MODELOS DE REFERÊNCIA 22

3.1.1. Modelo de Patterson 22

3.1.2. Modelo de Dewan 24

3.2. ESTILOS ARQUITETURAIS 26

3.2.1. MVC 26

3.3. CONSIDERAÇÕES FINAIS 30

4. METODOLOGIA 32

4.1. OBJETIVOS DA METODOLOGIA 33

4.1.1. Geral 33

4.1.2. Específicos 33

4.2. ANÁLISE DE COMPETIDORES 33

4.3. PROTOTIPAÇÃO 34

4.4. CENÁRIOS 35

5. ANÁLISE DE COMPETIDORES 37

5.1. RENDEZVOUS 37

5.2. NSTP 38

5.3. ANTS 39

5.4. DREAMTEAM 40

5.5. GROUPKIT 42

5.6. HABANERO 44

5.7. SHARE-KIT 45

5.8. COMPARAÇÃO ENTRE COMPETIDORES 46

5.9. REQUISITOS LEVANTADOS A PARTIR DA ANÁLISE DE COMPETIDORES 47
5.10. CONSIDERAÇÕES FINAIS 52
6. PROTOTIPAÇÃO 53
6.1. PROTÓTIPO DA PLATAFORMA DE G*ROUPWARE* PLATTUS 53
6.1.1. Arquitetura de Distribuição 53
6.1.2. Comunicação 53
6.1.3. Consciência de colaboração 54
6.1.4. Espaço compartilhado 55
6.1.5. Controle de sessão 55
6.1.6. Definição de papéis 56
6.1.7. Percepção 56
6.1.8. Componentes de interfaces (widgets) 56
6.1.9. Independência de plataforma 57
6.2. SERVIDOR PLATTUS 57
6.2.1. Mensagens 59
6.2.2. API de comunicação cliente/servidor 60
6.2.3. Exemplo de envio e recebimentos de mensagens 61
6.3. COMPONENTE COLABORATIVO GÉRARD 62
6.3.1. Estruturas aditivas 62
6.3.2. Protótipo do componente Gérard 65
6.4. CONSIDERAÇÕES FINAIS 68
7. ANÁLISE DE CENÁRIOS 69
7.1. PRIMEIRA ITERAÇÃO 69
7.1.1. Atores 69
7.1.2. Contexto 69
7.1.3. Ambiente (setting) 69
7.1.4. Roteiro (plot) 71
7.1.5. Roteiro positivo 77
7.1.6. Roteiro negativo 79
7.1.7. Requisitos levantados 83
7.2. SEGUNDA ITERAÇÃO 87

7.2.1. Protótipo 87
7.2.2. Atores 91
7.2.3. Contexto 91
7.2.4. Ambiente (setting) 91
7.2.5. Roteiro positivo 92
7.2.6. Roteiro negativo 94
7.2.7. Requisitos levantados 98
7.3. CONSIDERAÇÕES FINAIS 102
8. CONCLUSÕES E TRABALHOS FUTUROS 104
REFERÊNCIAS BIBLIOGRÁFICAS 106

LISTA DE FIGURAS

Figura 2.1 – Taxonomia da colaboração apoiada por computador.....16
Figura 3.1 – Exemplos da taxonomia de Petterson 23
Figura 3.2 – Arquitetura genérica de Dewan.....25
Figura 3.3 – Cenário de interação do MVC.....27
Figura 3.4 – Arquitetura Centralizada.....28
Figura 3.5 – Arquitetura Replicada. 28
Figura 3.6 – Arquitetura Distribuída. 29
Figura 3.7 – Arquitetura Híbrida. 30
Figura 4.1 – Alocação de técnicas de usabilidade aplicadas à análise. 32
Figura 4.2 – Fluxo de aplicação das técnicas selecionadas. 33
Figura 5.1 – CardTable: uma aplicação desenvolvida com Rendezvous 37
Figura 5.2 – MapTool: Ferramenta desenvolvida com ITCOLE architecture. 40
Figura 5.3 – Ambiente de execução do DreamTeam 41
Figura 5.4 – Plataforma DreamTeam rodando em dispositivos Palm. 42
Figura 5.5 – Aplicações desenvolvidas com Groupkit. 43
Figura 5.6 – Tela do cliente Habanero. 45
Figura 5.7 – Aplicação cliente do Share-Kit. 46
Figura 6.1 – Componente “Lista de Usuários”.....56
Figura 6.2 – Componente de *Chat*.....57
Figura 6.3 – Arquitetura simplificada de comunicação.....58
Figura 6.4 – Modelo de comunicação. 59
Figura 6.5 – Exemplo de diagrama e problema de composição. 63
Figura 6.6 – Exemplo de diagrama e problema de transformação. 63
Figura 6.7 – Exemplo de diagrama e problema de comparação.....64
Figura 6.8 – Exemplo de variações dos diagramas básicos.....64
Figura 6.9 – Estrutura aditiva composta. 65
Figura 6.10 – Interface do ambiente Gérard. 66
Figura 7.1 – Laboratório de informática.....70

Figura 7.2 – Telas de escolha do grupo e *login*. 72
Figura 7.3 – Leandra entra no ambiente 72
Figura 7.4 – Matheus entra no ambiente 73
Figura 7.5 – Leandra desenha o diagrama no ambiente. 74
Figura 7.6 – Leandra termina de desenhar o diagrama. 75
Figura 7.7 – Leandra e Matheus concluem a primeira tarefa. 76
Figura 7.8 – Leandra passa o controle para Matheus. 76
Figura 7.9 – Matheus recebe o controle do ambiente. 77

LISTA DE TABELAS

Tabela 5.1 – Comparação entre competidores..... 47
Tabela 6.1 – Botões da barra de ferramentas do Gérard 66
Tabela 7.1 – Requisitos levantados na análise de cenários (primeira iteração) 83
Tabela 7.2 – Requisitos levantados com a análise de cenários (segunda iteração)..... 99

RESUMO

Aprendizado Colaborativo Apoiado por Computador tem se firmado como um paradigma de ensino. Neste articulam-se diferentes conceitos de aprendizagem e interação em grupo. Os ambientes colaborativos síncronos visam facilitar a construção de conhecimento e o desenvolvimento de competências entre participantes de um grupo.

Componentes síncronos de aprendizagem (também chamados de sistemas de *groupware*) podem ser caracterizados por prover um conjunto de objetos virtuais compartilhados. Estes são manipuláveis por ferramentas e constituem-se num espaço compartilhado de interação entre usuários.

O problema que estamos focando consiste levantar, a partir da interação entre usuários, requisitos que colaborem com a definição de uma arquitetura de componentes colaborativos síncronos que resolva problemas de comunicação, consciência de colaboração, manutenção do espaço compartilhado, controle de acesso ao espaço compartilhado, definição de papéis e percepção (*awareness*).

Este trabalho utilizou técnicas de design para usabilidade com o intuito de resolver problemas relativos à interação mediada por componentes síncronos de aprendizagem, propondo um conjunto de requisitos para construção de novos sistemas que favoreçam a interação em grupo através de análises centradas nas necessidades dos usuários.

ABSTRACT

Computer Supported Collaborative Learning has become a teaching-learning paradigm. It deals with different concepts of learning and group interaction. Synchronous collaborative environments aim at facilitating the construction of knowledge and the development of skills among the participants of a group.

Synchronous learning components (also called groupware) can be characterized as providers of a set of shared virtual objects. Those objects can be manipulated through tools and form a shared space for user interaction.

The problem on which we are focusing is eliciting, from user interaction, requirements that help to define an architecture of synchronous collaborative components, which would solve problems in communication, collaboration consciousness, maintainability of the shared space, shared space access control, role definition and awareness.

This work made use of usability design techniques with the goal of solving problems related to interaction mediated by synchronous learning components, proposing a set of requirements to be applied to the construction of new systems which would favour group interaction through user needs-centered analysis.

1. INTRODUÇÃO

O aprendizado mediado por computador pode apresentar um ambiente, no qual os alunos interagem uns com os outros, colaborando para solucionar um dado problema. A principal promessa do aprendizado colaborativo suportado por computador, segundo Kumar [KUM1996], é permitir que os alunos aprendam em um ambiente relativamente realista, cognitivamente motivacional e socialmente rico.

Através de um componente síncrono de aprendizagem, um tipo específico de *groupware* síncrono, estudantes geograficamente dispersos, conectados via uma rede de computadores, podem colaborar em tempo real através de um espaço de trabalho compartilhado (*shared workspace*) [GUT1995].

A manutenção do espaço de trabalho compartilhado, ou seja, como manter objetos compartilhados com os quais os usuários interagem, tem sido a principal preocupação durante o design de sistemas de *groupware* [PHI1999]. Mas, com a evolução destes sistemas, as preocupações estão movendo-se do nível arquitetural (comunicação) para um nível mais próximo ao usuário (interação) [GUE2001].

Este trabalho utilizou técnicas de design para usabilidade, em particular uma análise de competidores e a técnica de análise de cenários, com o objetivo de, a partir da análise da interação entre usuários, levantar requisitos de *groupware* que colaborem com a definição de uma arquitetura de uma plataforma de apoio a aplicações colaborativas educacionais síncronas.

Observaram-se os principais problemas existentes no desenvolvimento de arquiteturas de *groupware* e propôs-se um conjunto de requisitos para construção de novos sistemas que favoreçam a interação em grupo apoiados na análise centrada no usuário.

No próximo capítulo, apresentamos uma revisão da literatura dos conceitos de aprendizado colaborativo apoiado por computador, destacando as

formas de colaboração, focando na colaboração síncrona e apresentando componentes síncronos de aprendizagem, que são um tipo específico de *groupware*. Este capítulo conclui com uma análise do principal problema dos sistemas de *groupware* e aponta para o problema tratado neste trabalho.

O terceiro capítulo continua a revisão da literatura, destacando as principais abordagens utilizadas no desenvolvimento de *groupware e* que foram importantes para a implementação do protótipo desenvolvido neste trabalho.

O quarto capítulo apresenta a metodologia utilizada no trabalho, cada uma das técnicas aplicadas no desenvolvimento deste trabalho: análise de competidores, prototipação e análise de cenários.

Nos três capítulos seguintes apresentam-se os resultados da análise de competidores (Capítulo 5), detalhes do protótipo da plataforma de *groupware* Plattus e do componente síncrono de aprendizagem Gérard (Capítulo 6) e os resultados empíricos da análise dos cenários de utilização do *software* Gérard (Capítulo 7).

Finalmente, o oitavo capítulo apresenta as conclusões evidenciadas a partir dos resultados obtidos e descritos nos três capítulos anteriores. Este capítulo apresenta também os trabalhos futuros relacionados com os resultados deste trabalho.

2. AMBIENTES EDUCACIONAIS SÍNCRONOS

Aprendizagem colaborativa envolve situações que contemplam um grupo de alunos trabalhando conjuntamente em um ambiente educacional, tanto em tarefas comuns ou através do uso de ferramentas computadorizadas, como por exemplo, um jogo eletrônico ou um componente síncrono de aprendizagem.

Aprendizagem colaborativa é uma idéia antiga na educação que tem recebido atenção significativa nas duas últimas décadas. Porém, ao invés de refletir a idéia inicial onde a cognição era vista como um produto individual, o contexto das interações sociais era visto apenas como pano de fundo das atividades individuais, o foco tem mudado para análise do grupo como objeto de estudo [DIL1996].

Hiltz [HIL1988] descreve que o processo de aprendizado colaborativo enfatiza a participação ativa, a interação entre os alunos, discussão e construção do conhecimento que emerge a partir de compartilhamento de idéias e informação.

2.1. APRENDIZADO COLABORATIVO APOIADO POR COMPUTADOR

A aprendizagem colaborativa apoiada por computador (*Computer-Supported Collaborative Learning* - CSCL) é a área de pesquisa que estuda o uso da tecnologia computacional no suporte às atividades da aprendizagem colaborativa [MAN1997]. É considerada uma área multidisciplinar que fundamenta-se na aplicação de conhecimentos derivados de diferentes campos de pesquisa, tais como a ciência da computação, a sociologia, antropologia, psicologia, pedagogia, ciências cognitivas, comunicação, entre outros [MEZ2002].

Riel [RIE1992] diz que embora os benefícios de um ambiente de aprendizagem colaborativa tenham sido estabelecidos, a transformação das salas de aulas tradicionais em organizações sociais que permitam aprendizagem colaborativa tem provado ser uma tarefa complexa. Adicionar o computador em uma tarefa não reduz sua complexidade. Esta área de pesquisa, entretanto, está tentando expandir as fronteiras da aprendizagem colaborativa através da utilização de computadores.

CSCL enquadra-se como subárea de *Computer Supported Cooperative Work*[1] (CSCW), a qual por sua vez é considerada uma subárea de HCI[2]. Na área CSCW inclui-se a investigação das chamadas aplicações de *groupware*[3]. Estas aplicações colaborativas visam dar suporte à interações de grupos, enquanto que a área de CSCL é mais voltada ao estudo deste tipo de aplicação abordando apenas com um domínio educacional [KOS1992].

2.1.1. TAXONOMIA DA COLABORAÇÃO APOIADA POR COMPUTADOR

Classificar de forma genérica como a colaboração pode acontecer mediada pelo computador é uma tarefa difícil devido à diversidade de aplicações e domínios deste tipo de tecnologia. Entretanto, uma categorização bastante aceita na literatura de CSCW e CSCL usa as dimensões espaço/tempo apresentadas por Ellis *et al.* [ELL1991] e também por Johansen [JOH1992].

Segundo Johansen [JOH1992], a colaboração apoiada por computador acontece em diferentes modos de interações. A combinação das dimensões de tempo e espaço estabelece quatro tipos de colaboração com características bem distintas. As interações podem ocorrer ao mesmo tempo (síncronas) ou em diferentes momentos (assíncronas). Os usuários podem encontrar-se no mesmo local (próximos) ou em lugares diferentes (dispersos). A Figura 2.1 mostra as diferentes alternativas para colaboração mediada por computador.

[1] Em Português: Trabalho cooperativo apioado por computador
[2] Do Inglês, *Human-Computer Interaction*, Interação Homem-Máquina foi adotada em meados dos anos 80 para descrever o campo emergente de estudo que foca em todos os aspectos da interação entre usuários e computadores.

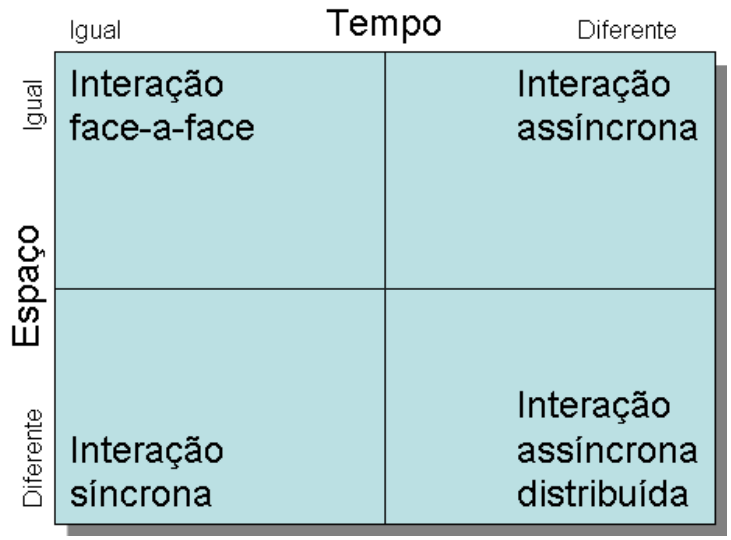


Figura 2.1 – Taxonomia da colaboração apoiada por computador.

Mesmo tempo/mesmo espaço refere-se a situações em que a sala de aula tradicional é provida com computadores para os alunos. Neste modo de interação o ambiente da sala de aula tradicional é melhorado com o uso de sistemas de grupos.

Mesmo tempo/espaços diferentes refere-se a situações nas quais as interações ocorrem de forma síncrona, mas os usuários encontram-se separados geograficamente. A comunicação ocorre através de vídeo, áudio ou mesmo de forma textual, por exemplo, através de um *chat*. Neste caso, a sala de aula não representa mais um lugar físico, mas sim, o conjunto de alunos que interagem através do sistema de grupo. A comunicação e o compartilhamento de informação são essenciais para manter o contexto compartilhado entre os usuários.

Tempo diferente/mesmo espaço refere-se a situações nas quais, dada a sua natureza própria, restringem consideravelmente a interação entre os seus utilizadores [ROB1998]. Neste caso, por exemplo, o professor poderia definir uma atividade colaborativa onde os alunos usariam apenas um determinado computador para criar um texto colaborativamente. Porém, os alunos usariam o computador em momentos diferentes, mas sempre no mesmo computador e deixariam no próprio documento os comentários para os demais colegas. Este

exemplo mostra que esta é uma forma de interação bastante difícil de ser encontrada.

Tempo diferente/espaços diferentes refere-se a situações onde as interações e as atividades são realizadas individualmente, servindo o sistema de mediador, permitindo a troca de informação e a coordenação de atividades. O correio eletrônico (*e-mail*) e os sistemas de listas de discussão constituem exemplos deste tipo de sistemas.

Este trabalho concentra-se na colaboração síncrona apoiada por computador, ou seja, àquelas que ocorrem ao mesmo tempo. Koschmann *et al.* [KOS1993] divide as aplicações que apóiam a colaboração síncrona em dois grupos: aplicações desenvolvidas para uso em sala de aula e aquelas desenvolvidas para comunicação entre salas de aulas via redes de longa distâncias (por exemplo, a Internet). Palmer [PAL1994] também apresenta uma classificação equivalente, que ele chama de colaboração co-presencial e distribuída.

Colaboração co-presencial caracteriza-se pela presença física dos participantes no mesmo local, por exemplo, em um laboratório onde os computadores estejam conectados em rede, mas a qualquer momento os alunos podem se comunicar face-a-face.

Colaboração distribuída ocorre quando várias pessoas trabalham em computadores separados, conectados por uma rede, e utilizando software especialmente projetado para utilizar conexões de rede.

Segundo Bricker [BRI1995], a colaboração co-presencial é preferível à colaboração distribuída. Mas, essa afirmação depende do paradigma instrucional que estiver sendo utilizado. Dentro do paradigma de educação à distância a colaboração distribuída torna-se muito mais valiosa.

Dentre as formas de colaboração apresentadas, este trabalho concentra-se nas colaborações síncronas educacionais, co-presenciais ou distribuídas, apoiadas por sistemas colaborativos conhecidos como sistemas de

groupware. Na próxima seção apresentamos o conceito e características deste tipo de sistema.

2.1.2. GROUPWARE SÍNCRONO DE APRENDIZAGEM

O conceito de *groupware* mais aceito foi definido por Ellis *et al.* [ELL1991]. Ellis *et al.* definem *groupware* como sistemas baseados em computador que suportam grupos de pessoas engajadas em uma tarefa comum e que provêem interface para um ambiente compartilhado. Este conceito é bastante genérico, pois considera um *groupware* qualquer sistema computacional que ofereça algum suporte para o trabalho em grupo, não importando se o sistema foi construído com essa finalidade.

A partir desta definição, *groupware* pode ser tanto síncrono como assíncrono, onde a comunicação é o principal aspecto deste tipo de sistema. Uma indicação disso é que no mesmo artigo considera-se *email* como um exemplo de *groupware*.

Nosso estudo focaliza nos sistemas de *groupware* síncronos que se propõem ao apoio de atividades educacionais. Nós chamamos estes sistemas de componentes síncronos de apredizagem, e utilizamos a definição de Gutwin [GUT1995] para definir este tipo de sistemas.

Componente síncrono de aprendizagem pode ser caracterizado por prover um ambiente que permite estudantes geograficamente dispersos ou co-presentes, conectados via uma rede de computadores, colaborarem em tempo real através de um espaço de trabalho compartilhado.

No presente trabalho estaremos utilizando o termo *groupware* como sinônimo de componentes síncronos de aprendizagem e por conseqüência atendendo a este conceito mais restrito que apresentamos.

É fácil perceber, a partir das diversas definições de *groupware* encontradas na literatura, que um aspecto chave dos sistemas de *groupware* é a manutenção do espaço de trabalho compartilhado. Em aplicações monousuárias a principal preocupação da interface do usuário é apresentar na tela, de forma mais

realista possível, o produto das ações do usuário. Esta estratégia é conhecida como WYSIWYG (*What You See Is What You Get*[1]).

Em sistemas de *groupware* a interface do espaço de trabalho é compartilhada com diversos usuários e existe a necessidade de manter o acoplamento entre o espaço de trabalho compartilhado e as interfaces dos usuários. As estratégias de acoplamento de interfaces derivam semanticamente da estratégia monousuária, mas são muitas vezes limitadas pela estratégia de manutenção do espaço compartilhado escolhida pelos projetistas do *groupware*.

As principais estratégias de acoplamento de interface utilizadas por sistemas de *groupware* são:

WYSIWIS (*What You See Is What I See*[2]) – Também conhecida como WYSIWIS estrita, apresenta a todos os usuários exatamente a mesma visão. Ideal para colaborações onde as atividades são altamente acopladas, como treinamento pessoal, demonstração do uso de um determinado software, ou certos tipos de atividades de tutoria. Não é recomendado para atividades fracamente acopladas, porque todos vêem o mesmo cursor e a posição de rolagem da tela é a mesma para todos.

WYSIWIS relaxada – Não exige que a visão seja exatamente a mesma para todos os usuários. Os usuários podem rolar livremente a tela de forma independente e realizar suas próprias operações como editar ou mover objetos, até que uma alteração no espaço compartilhado resulte na atualização da interface.

WYMIWIM (*What You Model Is What I Model*[3]) - Não exige que a interface seja exatamente a mesma para todos os usuários. Permite o uso de múltiplas representações, por exemplo, um aluno pode está vendo uma tabela, enquanto outro está vendo um gráfico, que representam os mesmos dados no espaço compartilhado.

[1] Em Português: O que você vê é o que você tem.
[2] Em Português: O que você vê é o que eu vejo.
[3] Em Português: O que você modela é o que eu modelo.

2.2. PROBLEMA

A manutenção do espaço de trabalho compartilhado, ou seja, como manter objetos compartilhados com os quais os usuários interagem, é a principal preocupação dos sistemas de *groupware*, mas não é a única. Com a evolução destes sistemas, as preocupações estão movendo-se do nível arquitetural para um nível mais próximo ao usuário.

O problema que estamos focando consiste em levantar, a partir da interação entre usuários, requisitos de *groupware* que colaborem com a definição de uma arquitetura de uma plataforma de apoio a aplicações colaborativas educacionais síncronas.

No próximo capítulo, apresentamos uma revisão das principais abordagens de arquiteturas de *groupware*. Elas procuram simplificar o desenvolvimento de sistemas de *groupware* através da proposta e codificação de estruturas apropriadas ao domínio do problema.

3. ARQUITETURAS DE *GROUPWARE*

Este capítulo continua a apresentação da revisão da literatura iniciada no capítulo anterior, porém focaliza nas principais abordagens utilizadas no desenvolvimento de *groupware e* que foram importantes para a implementação deste trabalho. Iniciaremos com a apresentação das gerações de sistemas de *groupware* e seguimos com os principais modelos de referência e estilos arquiteturais empregados no desenvolvimento deste tipo de sistema.

O desenvolvimento de *groupware* é uma tarefa complexa porque envolve aspectos que não aparecem no desenvolvimento de sistemas monousuários. Nas últimas décadas, autores têm trabalhado para melhorar a eficácia deste tipo de sistema. Segundo Guerrero & Fuller [GUE2001], isso gerou quatro gerações de aplicações:

A primeira geração é caracterizada pelo desenvolvimento dos primeiros sistemas colaborativos. Estes sistemas eram monolíticos, construídos em um único bloco lógico, não separando preocupações em camadas ou componentes diferentes. A linguagem C era predominante nesta geração. Estes sistemas preocupavam-se basicamente apenas com a comunicação entre os processos do sistema. Um exemplo de sistema desta geração é o Share-Kit [ROT1998].

A segunda geração é marcada pelo amadurecimento das ferramentas de apoio ao desenvolvimento de *groupware*. Representantes desta geração são o GroupKit [ROS1996] e o Habanero [NCS2005]. Esta geração caracteriza-se pelo surgimento de *toolkits* e extensões de linguagens que fornecem ao programador mecanismos que facilitam o desenvolvimento deste tipo de sistema.

A terceira geração caracteriza-se pela popularização das linguagens orientadas a objeto e ao surgimento de *frameworks* que encapsulam as funcionalidades de colaboração, como por exemplo, o NSTP (*Notification Service Transfer Protocol*) [PAT1996] [DAY1997]. Uma das vantagem em relação às gerações anteriores é o reuso de código garantido pelo paradigma de orientação a objetos.

Finalmente, a quarta geração distingue-se pela incorporação de objetos distribuídos (CORBA, DCOM, Enterprise Javabeans e RMI) na construção de *groupware* e arquitetura baseada em componentes. São representantes desta geração o ANTS [GAR2001, GAR2002] e o DreamTeam [ROT1998, ROT2000].

As dificuldades intrínsecas do desenvolvimento de *groupware* levaram a uma vasta gama de abordagens de desenvolvimento. Estas arquiteturas podem ser mapeadas conceitualmente a partir dos modelos de referência e estilos arquiteturais propostos para os sistemas de *groupware*.

3.1. MODELOS DE REFERÊNCIA

Os modelos de referência tentam descrever a estrutura de um sistema de *groupware* completo em um nível abstrato. Modelos de referência podem ser vistos tanto como prescritivos como descritivos [PHI1999]. Do ponto de vista prescritivo eles expressam uma perspectiva filosófica de como os sistemas de *groupware* podem ser construídos, enquanto que na visão descritiva eles podem prover um *framework* canônico, com o qual se pode discutir sobre a estrutura de sistemas de *groupware* existentes.

Nesta seção, apresentamos dois modelos que são relevantes para este trabalho: os modelos de Patterson e de Dewan.

3.1.1. MODELO DE PATTERSON

O primeiro modelo de referência desenvolvido especificamente para sistemas de *groupware* apareceu na taxonomia proposta por Patterson [PAT1995], motivado pela observação de que o principal desafio das aplicações de grupo síncronas é a manutenção do ambiente compartilhado. O modelo de Patterson divide o estado da aplicação em quatro níveis: o primeiro é o *display state*, que é implementado no *hardware* de vídeo que controla a tela do monitor do usuário; o segundo é o *view state*, a representação visual lógica da estrutura de dados interna da aplicação; o terceiro é o *model state*, a estrutura de dados interna que representa o ambiente compartilhado e o quarto é o *file state*, a representação persistente do modelo.

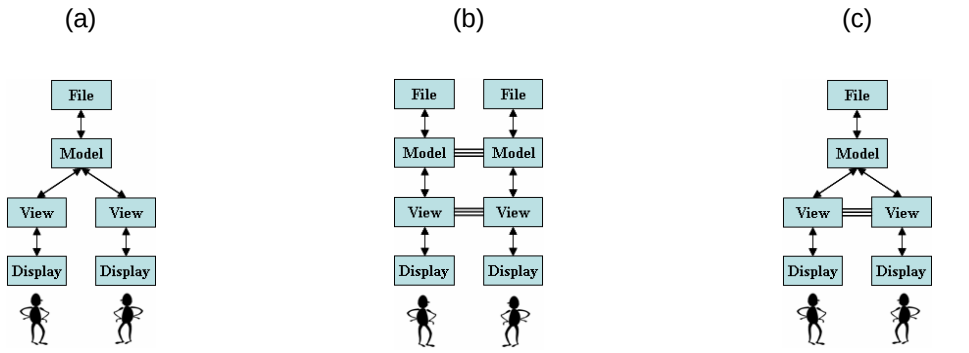


Figura 3.1 – Exemplos da taxonomia de Petterson

A taxonomia de Patterson, ilustrada na Figura 3.1, propõe três classes de arquitetura de *groupware*: aquelas baseadas no compartilhamento do estado, aquelas baseadas na replicação do estado com sincronização[1] e as híbridas, que incluem as duas abordagens anteriores. O benefício chave do compartilhamento do estado da aplicação é a simplicidade, enquanto que o benefício da replicação do estado é o melhoramento do desempenho e a capacidade de desligar e ligar a sincronização.

Na Figura 3.1(a) é apresentada a arquitetura de um sistema de *groupware* baseado no compartilhamento do modelo. Essa permite aos usuários ter diferentes visões dos dados da estrutura interna do modelo. O uso de diferentes componentes de visões, permite os mesmos dados sejam apresentados a usuários diferentes de formas diferentes. Por exemplo, alguém pode está vendo os dados em forma de tabela, enquanto outro os vê em forma de gráfico.

A Figura 3.1(b) mostra a arquitetura de um sistema compartilhado usando a arquitetura sincronizada. Os elementos sincronizados devem conter exatamente a mesma informação e são replicados por desempenho. Neste exemplo os usuários devem ter exatamente a mesma visão em suas telas. Isto

[1] A sincronização freqüentemente é garantida com a troca de mensagens de sincronismo. Estas mensagens são enviadas para todas as instâncias remotas da aplicação, garantindo que a cada alteração de um componente replicado suas cópias sejam atualizadas.

está relacionado com o modelo de acoplamento WYSIWIS estrito. Patterson sugere que onde as visões são sincronizadas, torna-se também necessário sincronizar separadamente os modelos.

O exemplo na Figura 3.1(c) é o de uma arquitetura híbrida, onde o modelo é compartilhado e as visões podem ser sincronizadas. Esta arquitetura provê um mecanismo simples para garantir a consistência do estado do modelo e permitir que a sincronia de visões possa ser ativada e desativada pelo próprio usuário. Isso permite, por exemplo, que um professor em uma apresentação à distância possa ativar ou desativar a sincronização das visões. A sincronização das visões, neste caso, resulta na situação onde as telas de todos os alunos seriam exatamente iguais (estariam sincronizadas) à tela do professor, garantindo o acompanhamento da apresentação.

A taxonomia de Patterson oferece uma representação simples das arquiteturas possíveis para sistemas de *groupware* e abstrai como alguns aspectos computacionais são alcançados, como por exemplo, concorrência e distribuição. Mas, a idéia é a simplificação do modelo, mantendo-se no nível conceitual favorecendo um melhor entendimento deste tipo de aplicação.

3.1.2. MODELO DE DEWAN

Este modelo foi definido a partir da arquitetura genérica de Dewan, que pode ser vista como uma generalização da taxonomia de Patterson [ROT2000]. A arquitetura de Dewan (Figura 3.2) é dividida em camadas, mas não define a responsabilidade de cada uma e nem determina o número máximo de camadas. O fluxo de dados é modelado através de eventos, que podem ser transferidos entre camadas horizontalmente ou verticalmente.

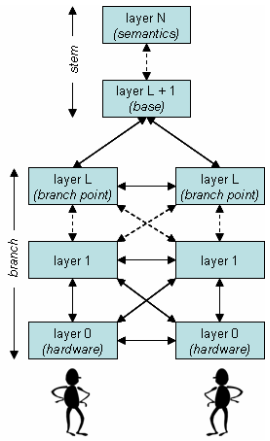


Figura 3.2 – Arquitetura genérica de Dewan.

Os objetos próximos ao usuário (camadas inferiores) são chamados de objetos de interação (*interatores)* das abstrações da camada superior. Um objeto no meio da hierarquia será tanto um objeto de interação quanto uma abstração. Um objeto de interação cria uma apresentação de sua abstração, a qual pode incluir transformações da abstração e informações adicionais. A comunicação entre os objetos no modelo acontece via eventos e pode ser síncrona ou assíncrona.

O modelo de Dewan, similarmente ao modelo anterior, oferece uma representação simples das arquiteturas possíveis para sistemas de *groupware*. Mas apresenta um conceito muito mais aceito pelos especialistas em Rede de Computadores, que é a organização e separação de funcionalidades em múltiplas camadas. Onde as camadas mais próximas ao usuário apresentam um maior nível de interação.

Como dito anteriormente, os modelos de referência estão em um nível abstrato, servindo mais para descrever e classificar sistemas de *groupware* que para construí-los. A seguir apresentaremos estilos arquiteturais que propõem modelos mais concretos de componentes, topologia e conectores, que podem ser utilizados para o desenvolvimento de novos sistemas de *groupware*.

3.2. ESTILOS ARQUITETURAIS

Um estilo arquitetural sugere um vocabulário de componentes (responsáveis pelo comportamento do sistema), tipos de conectores (interligando os componentes), uma topologia de interconexão (possibilitando vários tipos de interação e compartilhamento de informações entre esses componentes) e uma estratégia de controle de fluxo [MEN2002].

Phillips [PHI1999] declara que a questão chave que os desenvolvedores devem considerar, quando selecionam um estilo arquitetural para aplicações de *groupware*, é como construir sistemas que permitam vários usuários interagirem concorrentemente uns com os outros e com os dados compartilhados.

A maioria dos estilos arquiteturais encontrados em sistemas de *groupware* segue uma filosofia onde os dados da aplicação (o modelo) são mantidos separados do código da interface (a visão). O estilo arquitetural *Model-View-Controler* (MVC) é baseado nesta filosofia e é o mais comumente usado no desenvolvimento de sistemas de *groupware* [PHI1999]. O protótipo desenvolvido neste trabalho também vai seguir a filosofia MVC.

3.2.1. MVC

O estilo arquitetural MVC (Modelo / Visão / Controlador) foi introduzido com a linguagem Smalltalk [KRA1988] para prover uma separação eficaz entre a interface do usuário e a semântica da aplicação.

Nesta seção apresentaremos o MVC como foi definido no Smalltalk-80. O MVC é descrito também por Gamma *et al.* [GAM1994] como um exemplo de utilização do padrão de projeto *Observer*. A estrutura do MVC está ilustrada na Figura 3.3. A estrutura básica do MVC consiste de um modelo (*model*), que representa os dados da aplicação; um controlador (*controller*), que interpreta as entrada do usuário; e uma visão (*view*) que representa a saída.

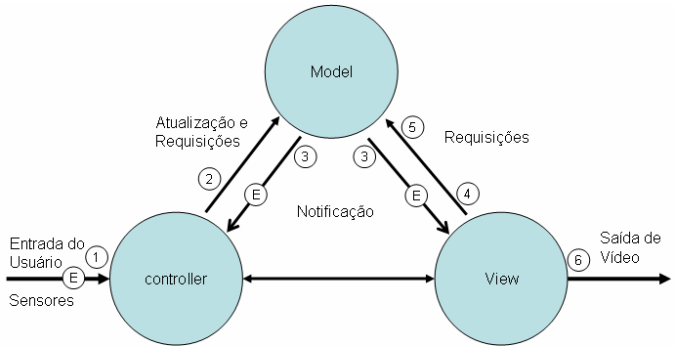


Figura 3.3 – Cenário de interação do MVC.

A Figura 3.3 também mostra um fluxo típico de interação via MVC, que começa com a entrada do usuário ①, o qual é interpretada pelo controlador e resulta na atualização do modelo ②. O modelo envia um evento de notificação para a visão e o controlador ③ informando que algum aspecto de seu estado foi alterado. A visão consulta o modelo para determinar o que foi alterado ④ e recebe os detalhes ⑤ que utiliza para atualizar a tela ⑥. O controlador pode também reagir à notificação alterando seu modo de interação.

Quando aplicado ao desenvolvimento de *groupware,* o MVC resolve o principal desafio das aplicações colaborativas síncronas: a manutenção do ambiente compartilhado, através da distribuição de seus componentes. A seguir analisamos as arquiteturas de distribuição dos componentes do modelo MVC apresentada por Suthers [SUT2001].

3.2.1.1. Centralizada

Esta arquitetura de distribuição provê somente uma aplicação completa e distribui cópias da GUI (*view* e *controler*) entre as máquinas clientes[1] (Figura 3.4). O modelo, a visão e o controlador da aplicação situam-se em um servidor. Este tipo de arquitetura apresenta conseqüentemente acoplamento do tipo WYSIWIS estrito.

[1] Máquinas clientes são computadores que participam da colaboração, assumindo o papel de cliente na arquitetura cliente/servidor.

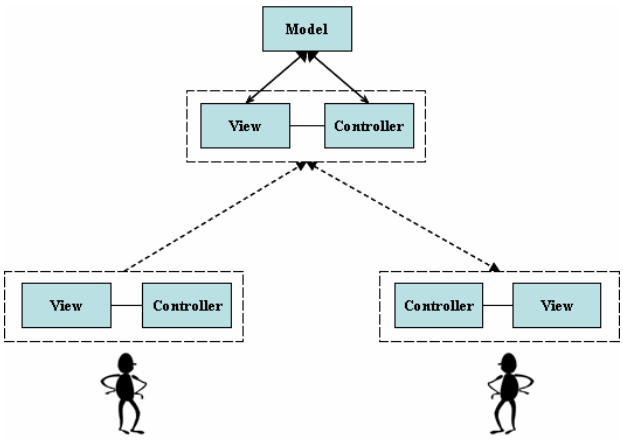


Figura 3.4 – Arquitetura Centralizada.

Como vantagem, esta arquitetura pode ser usada para compartilhar aplicações que não foram desenvolvidas com esta característica, através da captura e difusão dos eventos do sistema de janelas entre as máquinas clientes. Como restrição, apenas uma pessoa pode usar o mouse e o teclado em um determinado momento. É mais indicada para treinamento pessoal, como a demonstração do uso de um determinado software.

3.2.1.2. Replicada

Esta arquitetura de distribuição (Figura 3.5) é caracterizada por ter todos os três componentes MVC – modelo, visão e controle – replicados em todas as máquinas clientes, ou seja, a aplicação completa é executada em cada cliente e é provido algum tipo de sincronização entre elas.

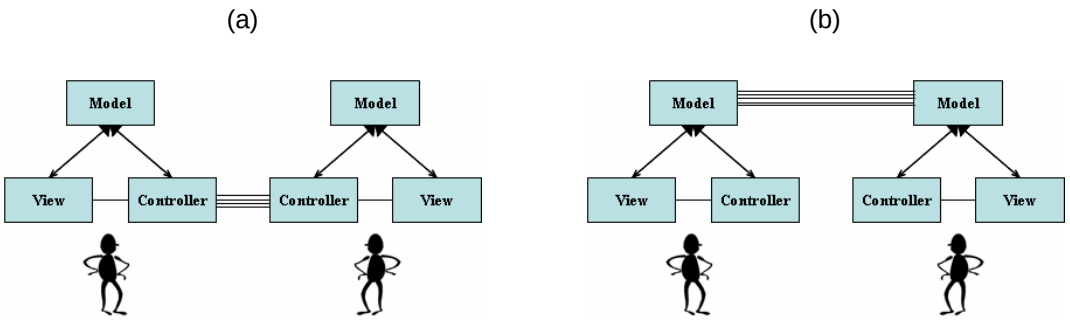


Figura 3.5 – Arquitetura Replicada.

Como vantagem, melhora a utilização dos recursos de rede, pois transmite apenas dados relacionados com os eventos dos controladores. Também cada cliente pode ser utilizado isoladamente sem colaboração.

A sincronização baseada em eventos do controlador (Figura 3.5a) pode não ser adequada para muitos tipos de sistemas de *groupware* educacionais, porque esta arquitetura está naturalmente associada ao tipo de acoplamento WYSIWIS estrito.

Já a sincronização baseada nos modelos (Figura 3.5b) permite acoplamento do tipo algum tipo de WYSIWIS relaxado, o que não obriga que todos os alunos estejam na mesma posição e vejam a mesma coisa.

3.2.1.3. Distribuída

Esta arquitetura é caracterizada pela distribuição dos componentes MVC entre diversos computadores (Figura 3.6). O modelo é mantido em um servidor e é compartilhado com todos os clientes, os quais possuem seus próprios controladores e visões.

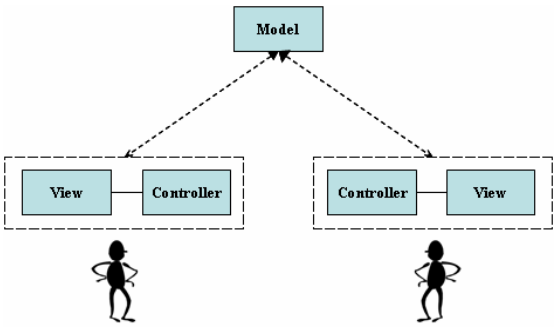


Figura 3.6 – Arquitetura Distribuída.

A arquitetura distribuída requer que somente eventos de atualização do modelo sejam enviados pela rede, tornando o uso dos recursos de rede mais eficiente. O acoplamento acontece no nível do modelo, podendo suportar WYMIWIM. Assim os usuários podem colaborar sobre o mesmo modelo, mas utilizando visões completamente diferentes.

Este tipo de arquitetura de distribuição tem a desvantagem de apenas poder ser utilizada em rede, não podendo ser executada em modo monousuário. Do ponto de vista educacional esta arquitetura permite o uso de múltiplas representações, por exemplo, um aluno pode estar vendo uma tabela, enquanto outro está vendo um gráfico, que representam os mesmos dados.

3.2.1.4. Híbrida

Nesta arquitetura são integradas características das arquiteturas replicada e distribuída (Figura 3.7), capturando o melhor dos dois mundos, gerando um modelo híbrido onde as aplicações podem ser utilizadas em modo monousuário e em modo colaborativo.

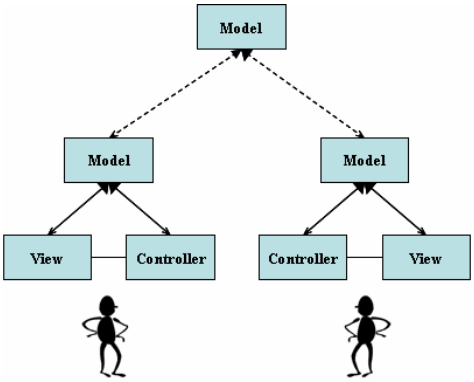


Figura 3.7 – Arquitetura Híbrida.

As aplicações podem executar em modo monousuário utilizando seu próprio modelo e salvando seu estado em um arquivo local, ou podem conectar-se com o servidor que provê armazenamento persistente favorecendo atualização WYMIWIM entre os clientes ativos.

Do ponto de vista educacional, aplicações construídas com este modelo permitem que os alunos possam utilizar as ferramentas individualmente para praticar antes de conectar com outros colegas e trabalhar colaborativamente.

3.3. CONSIDERAÇÕES FINAIS

Adotaremos a arquitetura híbrida, que apresenta características unificadas das arquiteturas replicada e distribuída, capturando o melhor dos dois

mundos, gerando um modelo híbrido onde as aplicações podem ser utilizadas tanto em modo monousuário como em modo colaborativo.

A manutenção do espaço compartilhado do protótipo seguirá a arquitetura com estado compartilhado, permitindo o acoplamento do tipo WYSIWIS relaxado. Este acoplamento acontece no nível do modelo, assim os usuários podem colaborar sobre o mesmo modelo, mas utilizando visões completamente diferentes.

No próximo capítulo apresentamos a metodologia utilizada no desenvolvimento deste trabalho, que utilizou técnicas de design de usabilidade para resolver problemas relativos à interação em componentes síncronos de aprendizagem.

4. METODOLOGIA

A análise de requisitos é uma das mais importantes fases do desenvolvimento de software e muito tem se escrito sobre técnicas para garantir o levantamento eficaz de requisitos. À exemplo disso, Ferre [FER2003] (Figura 4.1) apresenta um mapeamento entre diversas técnicas de usabilidade e suas aplicações nas atividades de análise de requisitos.

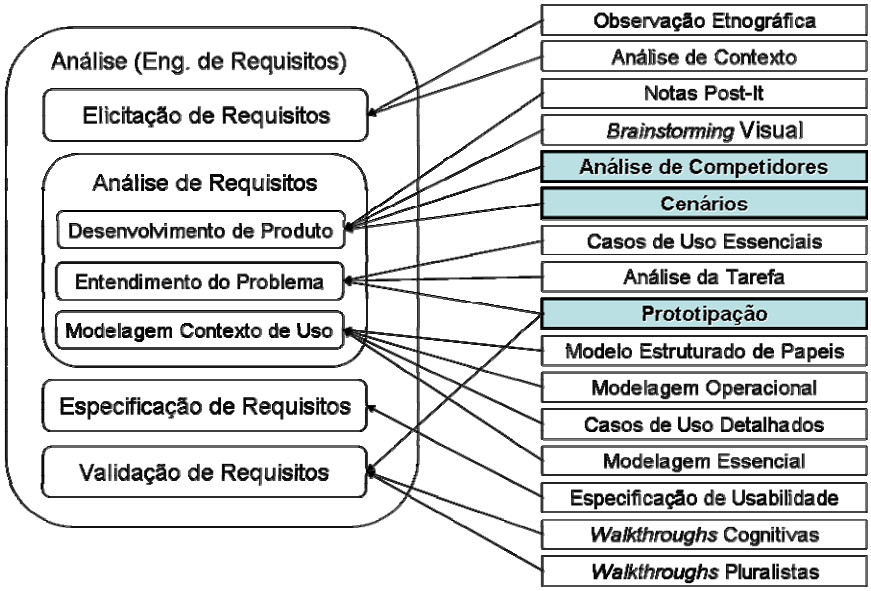


Figura 4.1 – Alocação de técnicas de usabilidade aplicadas à análise.

Dentre as diversas técnicas disponíveis para colaborar com a análise de requisitos, escolhemos três para compor nossa metodologia.

A Figura 4.2 apresenta como cada uma das técnicas selecionadas foi aplicada na metodologia. Na primeira fase foi aplicada a metodologia de análise de competidores com o objetivo de levantar os requisitos básicos dos sistemas de *groupware* já existentes. Uma vez levantados os requisito básicos, estes foram implementados em um protótipo que posteriormente foi utilizado na construção de cenários que ajudaram a levantar novos requisitos para o sistema.

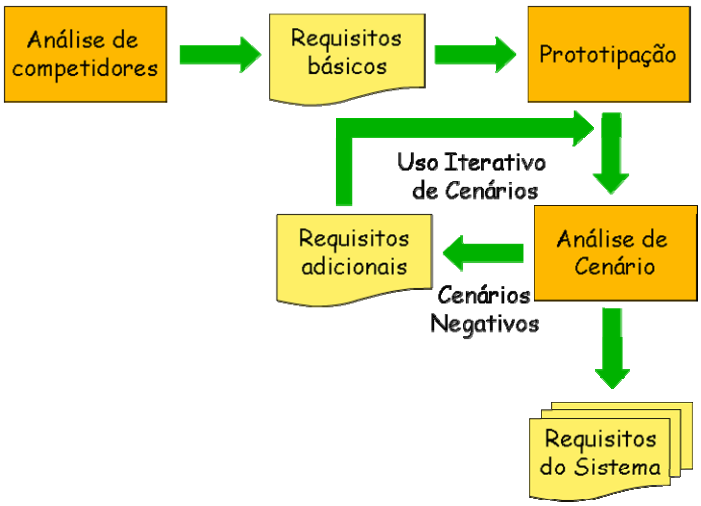


Figura 4.2 – Fluxo de aplicação das técnicas selecionadas.

A seguir, apresentamos em detalhes como cada uma das técnicas foi aplicada no desenvolvimento deste trabalho.

4.1. OBJETIVOS DA METODOLOGIA

4.1.1. GERAL

Levantar requisitos que favorecem a interação em grupo a serem incorporados em uma plataforma de *groupware*, visando o apoio ao desenvolvimento de componentes educacionais síncronos.

4.1.2. ESPECÍFICOS

Entender e levantar os requisitos da interação síncrona que são tratados em plataformas contemporâneas.

Implementar um protótipo de uma plataforma de *groupware*, que incorpore os requisitos elicitados;

Especificar requisitos a partir da geração de cenários de uso, baseados em resultados de experimentos com usuários.

4.2. ANÁLISE DE COMPETIDORES

A análise de competidores é uma técnica de HCI que consiste em examinar produtos existentes para coletar *guidelines*, princípios e boas práticas de

design para construção de um novo sistema [BOR2000]. A análise de competidores é útil para:

- Identificar as melhores práticas;
- Levantar requisitos;
- Avaliar os pontos fortes e fracos de nossos competidores;
- Reutilizar experiências de design; e
- Colaborar com a especificação funcional de novos sistemas.

Através do exame dos produtos competidores, podemos identificar as características, funcionalidades e elementos de projeto e serviços oferecidos pelos competidores aos seus usuários. Uma vez identificados, esses elementos podem ser reaplicados em novos produtos ou podem ser evitados se a experiência do competidor for negativa.

A análise de competidores foi utilizada neste trabalho para ajudar a entender e levantar os requisitos da interação síncrona que estão incorporados nas plataformas de *groupware* síncronos encontradas na literatura com o objetivo de incorporar as boas práticas de design e evitar os problemas já identificados por estas plataformas.

4.3. PROTOTIPAÇÃO

A elaboração de protótipos intermediários é uma técnica muito comum na literatura, principalmente usado na fase de levantamento de requisitos [PAD2003]. Protótipos aumentam a eficácia das atividades de levantamento de requisitos e a qualidade dos requisitos resultantes. Essa técnica teve como objetivo a exploração dos aspectos críticos dos requisitos levantados na análise de competidores.

Como a plataforma de *groupware* destina-se a oferecer uma infra-estrutura de colaboração, fez-se necessária a concepção de uma ferramenta colaborativa que utilizasse serviços oferecidos pela plataforma e fornecesse serviços perceptíveis e usáveis pelo usuário final.

4.4. CENÁRIOS

A análise de cenário é uma metodologia que colabora para elicitação de requisitos de um sistema. Análise de cenário é o processo de compreensão, análise e descrição do comportamento do sistema durante uma forma particular esperada de uso do sistema [HSI1994]. Sua descrição vívida da atividade humana promove reflexão focada sobre a utilidade e usabilidade de uma intervenção de projeto visionária [CAR1998]. O cenário pode funcionar como um protótipo “leve” [CAR2000], pois se constitui numa visão concreta e não apenas uma lista de requisitos do sistema, que pode ser apresentado ao usuário, expondo o projeto a sugestões e críticas.

Para facilitar o surgimento dos requisitos, foram construídos cenários caricaturados positivos e negativos (“*plus and minus scenarios*” [BØD2000]). Isto favoreceu o surgimento de novas demandas para a futura aplicação. O uso de cenários caricaturados ajuda a capturar melhor as vantagens e problemas do sistema proposto.

Neste trabalho apresentamos como requisitos podem ser desenvolvidos a partir da metodologia de análise de cenários e mais especificamente como os cenários caricaturados podem ajudar a refinar nossa visão inicial do sistema, destacando os pontos positivos e negativos de nossa solução.

Aplicamos a metodologia de cenários de forma iterativa neste trabalho, Os cenários criados abordam a colaboração entre alunos para realização de tarefa compartilhada apoiada por uma ferramenta síncrona colaborativa construída sobre nossa plataforma de *groupware*, Plattus.

Usamos cenários a partir de um conjunto de objetivos de alto nível, que fazem sentido para o usuário, através da manipulação da interface da ferramenta, mas que nos ajudaram a levantar requisitos que devem ser embutidos dentro da plataforma de *groupware* e que apoiarão a construção de novas ferramentas colaborativas.

Na primeira iteração os cenários, apesar de fictícios, basearam-se no trabalho de mestrado de Vânia Alves [ALV2005], no qual a autora observou vários

usuários usando a ferramenta Gérard, construída a partir do protótipo da plataforma Plattus. Assim, a primeira iteração reflete uma série de situações reais experimentadas por vários usuários, no uso da ferramenta Gérard, compiladas em uma única história hipotética.

A segunda iteração utiliza uma história totalmente fictícia, retratando um cenário futuro, pois ao invés de implementar um novo protótipo real para fazer testes de usabilidade, o novo cenário funciona com um protótipo "leve" (*"soft" prototype* [CAR2000]), onde o cenário apresenta um novo sistema, já incrementado com novas funcionalidades que buscam suprir os requisitos levantados na primeira iteração. Os elementos do futuro sistema aparecem no cenário embutidos nas interações do usuário. A partir deste novo cenário, baseado no novo sistema retratado por ele, um novo conjunto de requisito foi levantado, apresentando necessidade de novas funcionalidades ou refinando os requisitos levantados na iteração anterior.

5. ANÁLISE DE COMPETIDORES

Neste capítulo, apresentaremos os resultados da análise de competidores que teve como objetivo identificar as características, funcionalidades e elementos de projeto e serviços oferecidos pelos produtos competidores.

5.1. RENDEZVOUS

Rendezvous [PAT1990] [PAT1991] é uma arquitetura para criação de aplicações síncronas multiusuárias. Ela é composta de duas partes distintas, uma arquitetura de execução (*runtime*) para o gerenciamento de sessões multiusuárias e a arquitetura de acionamento (*start-up*) para gerenciamento de conexão inicial dos usuários (*login*).

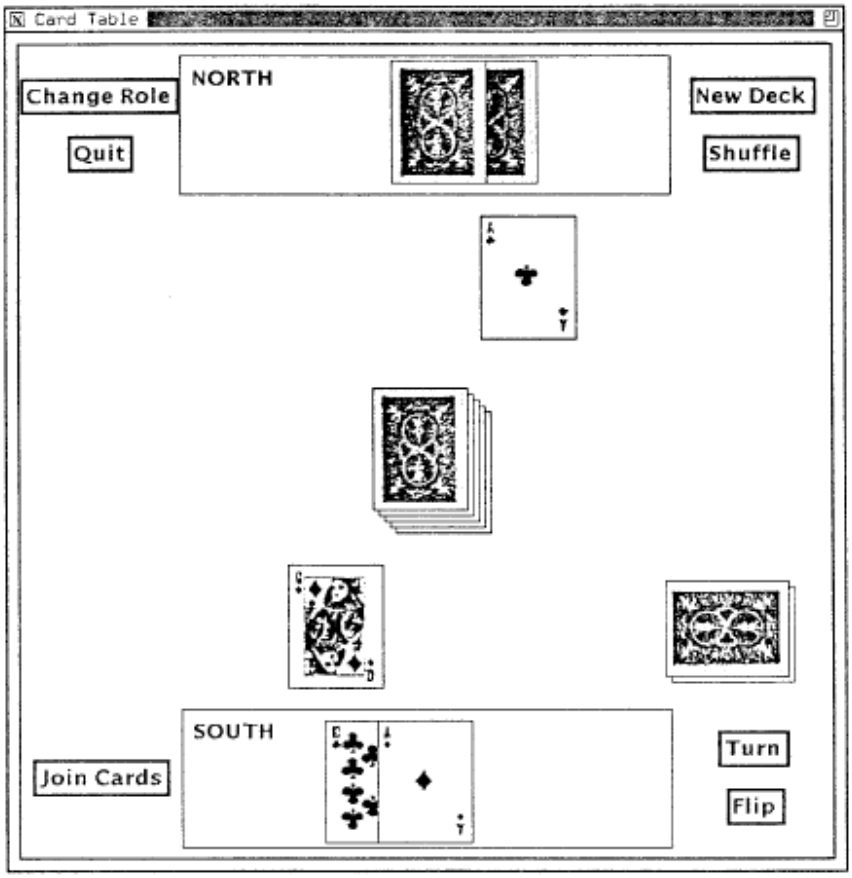


Figura 5.1 – CardTable: uma aplicação desenvolvida com Rendezvous

O modelo de comunicação do Rendezvous é centralizado no servidor chamado RESERV, que intermedia toda a comunicação. Os clientes utilizam

terminais virtuais que se comunicam com um processo centralizado que controla a aplicação.

A manutenção do estado compartilhado no Rendezvous é garantida através de restrições (*constraints*) que garantem a atualização de objetos em três dimensões: compartilhamento de objetos básicos (*underlying objects*), compartilhamento de visões (*views*) e compartilhamento de acesso.

Uma famosa aplicação construída pelo Rendezvous foi a *CardTable* (Figura 5.1). *CardTable* é uma aplicação multiusuária que permite vários usuários dispersos geograficamente interagirem simultaneamente com uma mesa de cartas virtual. A *CardTable* não é um jogo específico de cartas e sim uma simulação de um ambiente físico provido quando vários jogadores sentam ao redor de uma mesa para jogar cartas, as regras do jogo devem ser definidas pelos jogadores.

5.2. NSTP

NSTP (*Notification Service Transfer Protocol*) [PAT1996] [DAY1997] é um protocolo desenvolvido com a preocupação da manutenção da consistência entre aplicações compartilhadas. NSTP provê um serviço centralizado para construção de sistemas de *groupware* síncronos. Neste protocolo, os clientes trocam mensagens com o servidor de notificações (*Notification Server*), que tem o propósito de compartilhar o estado da aplicação entre um conjunto de clientes que trabalham juntos.

NSTP tem quatro tipos de objetos: lugares (*places*), coisas (*things*), fachadas (*facades*) e o próprio servidor de notificação. *Place* representa o espaço que é compartilhado pelos clientes da aplicação. *Things* são os objetos dentro do espaço compartilhado que podem ser manipulados pelos clientes e *Facades* são visões públicas do espaço compartilhado.

O protocolo do NSTP define ama arquitetura de colaboração e o conjunto de regras que rege a comunicação entre os clientes. O NSTP foi implementado em Java e C++. A implementação em Java, chamada *PlaceHolder*, está disponível sem qualquer custo para usuários não comerciais.

Um conceito interessante do NSTP é que ele prevê *browsers* que permitem aos usuários navegarem através das fachadas dos lugares (*places*) e a possibilidade de existirem objetos dentro dos ambientes que são *links* para outros lugares.

5.3. ANTS

ANTS [GAR2001] [GAR2002] é um *framework* colaborativo para desenvolvimento de aplicações multiusuárias. Baseado totalmente na tecnologia J2EE de Java, o ANTS utiliza o modelo de componentes JavaBeans para simplificar o desenvolvimento de aplicações colaborativas.

O modelo de comunicação do ANTS é centralizado, baseado em um serviço de mensagens. A camada de comunicação funciona como uma camada de adaptação para diferentes serviços de *middleware*. O canal de comunicação, chamado de *Collaboration BUS*, destina-se a propagação do estado compartilhado entre as aplicações e ao monitoramento de eventos pelo componente de percepção (*awareness*).

O ANTS apresenta um modelo bem elaborado de captura de informações de percepção sobre as ações dos usuários. O componente de percepção, chamado de AWS, é baseado no modelo sensor/mediador/atuador. Sensores podem ser associados a determinados eventos (mensagens) e o mediador liga um sensor a um ou mais atuadores. Assim que ocorre um evento o mediador dispara todos os atuadores ligados ao sensor. Os atuadores podem, por exemplo, armazenar os dados em banco de dados, mandar um *email* ou enviar mensagens para os usuários. Isso permite um sistema bastante extensível e flexível para provimento e armazenamento de informações de percepção.

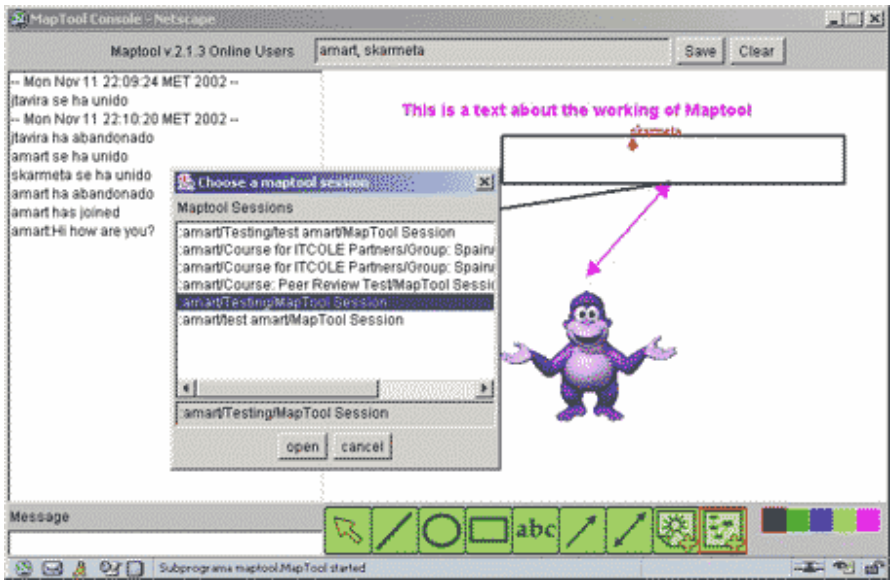


Figura 5.2 – MapTool: Ferramenta desenvolvida com ITCOLE architecture.

A arquitetura bem projetada do ANTS permitiu que alguns projetos fossem construídos a partir de sua estrutura como o JLE [GAR2001], MOVE [GAR2002], AORTA [ORO2004] e ITCOLE[1] *architecture* (Figura 5.2).

5.4. DREAMTEAM

O DreamTeam [ROT1998] [ROT2000] é um ambiente que permite o desenvolvimento de aplicações cooperativas como se fossem aplicações monousuárias, sem a preocupação com detalhes de rede. O ambiente é composto de três componentes: um ambiente de desenvolvimento, um ambiente de execução (Figura 5.3) e um ambiente de simulação.

DreamTeam baseia-se em uma arquitetura de comunicação descentralizada, onde não há necessidade de um servidor para gerenciar a comunicação entre os usuários. Isto traz uma vantagem de não se ter um ponto de gargalo no sistema, mas apresenta um modelo muito mais complexo de gerenciamento do estado compartilhado da aplicação.

[1] http://ants.dif.um.es/cscl/

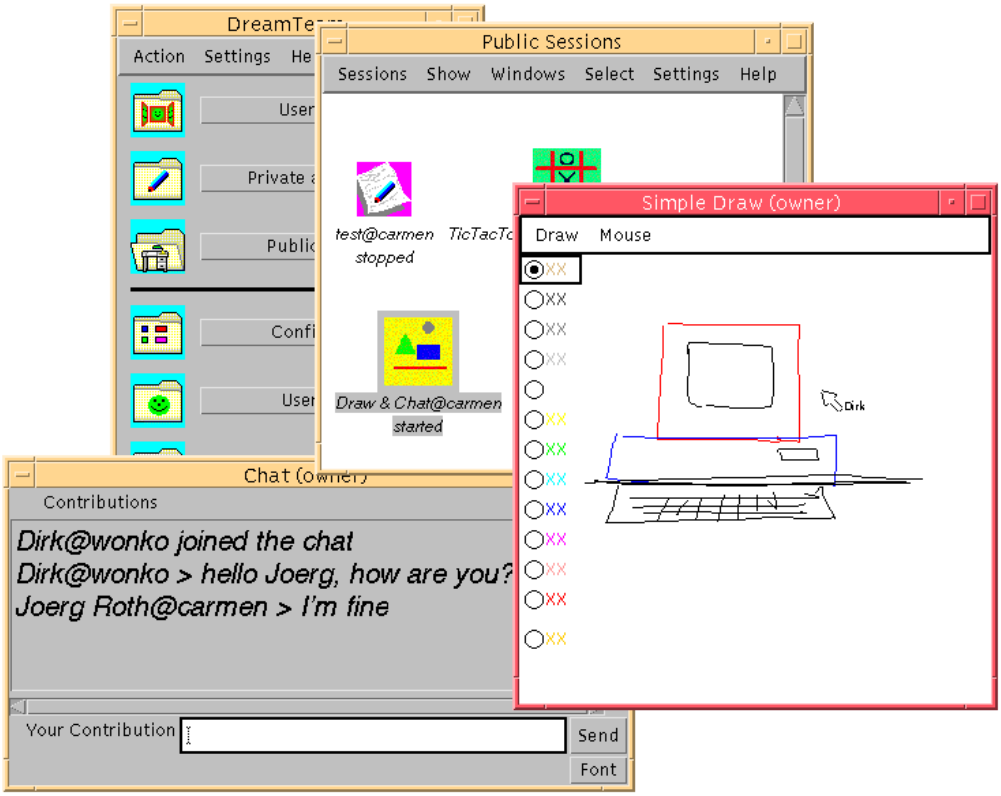


Figura 5.3 – Ambiente de execução do DreamTeam

Devido ao modelo de comunicação descentralizado o DreamTeam possui também um gerenciamento de sessão dos usuários descentralizado. A sessão é gerenciada pelo computador do usuário que cria a sessão e os outros usuários podem associar-se à sessão. Quando o usuário responsável pela criação da sessão sai, os outros usuários continuam engajados na sessão, mas nenhum novo membro pode entrar.

Figura 5.4 – Plataforma DreamTeam rodando em dispositivos Palm.

Dos ambientes pesquisados, o DreamTeam é o único que oferece um ambiente de simulação que permite testar os efeitos da rede no desempenho das aplicações compartilhadas. O ambiente simula atrasos de rede, falhas de comunicação, etc., permitindo ao desenvolvedor testar o uso dos aplicativos compartilhados em situações adversas [ROT2000].

O DreamTeam também é o único dos competidores analisados que apresenta uma versão do ambiente que é compatível com dispositivos portáteis (Figura 5.4).

5.5. GROUPKIT

Groupkit [ROS1996] é um pacote (*toolkit*) para desenvolvimento de aplicações distribuídas, desenvolvido na linguagem TCL-TK. O Groupkit oferece serviços padronizados para *groupware*, como gerenciamento de sessão, comunicação e interface de usuário distribuída.

Groupkit é usado para construir aplicações colaborativas síncronas (Figura 5.5), tais como ferramentas de desenho, editores de texto compartilhados, sistemas de reuniões, que podem ser utilizados por vários usuários simultaneamente.

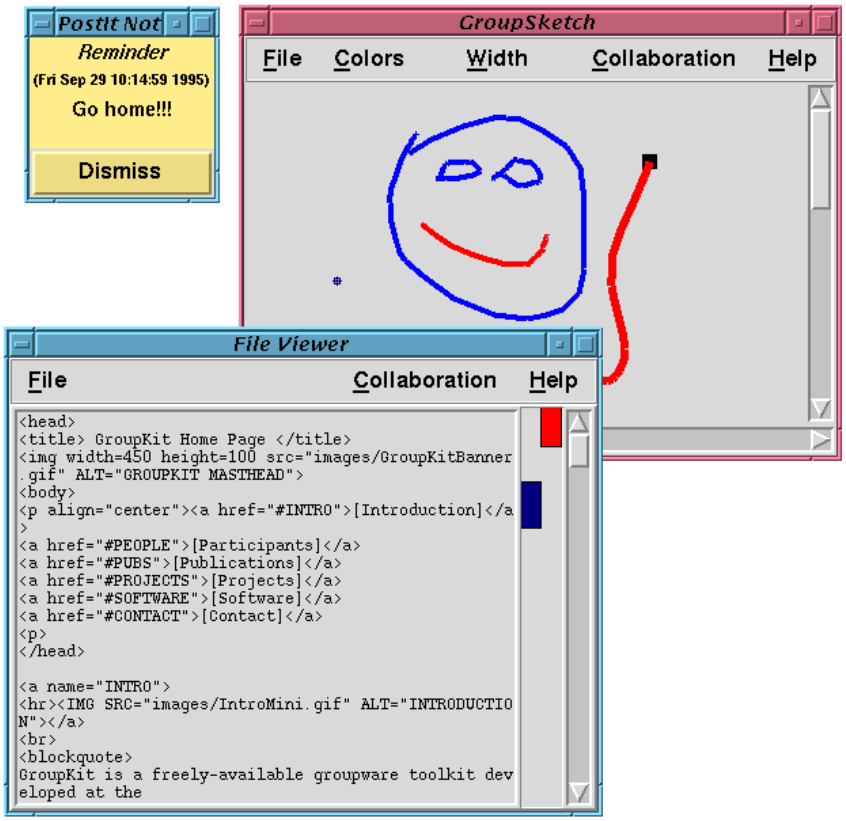


Figura 5.5 – Aplicações desenvolvidas com Groupkit.

O modelo de comunicação utilizado pelo Groupkit para manutenção do estado compartilhado das aplicações clientes é descentralizado. Porém, durante a fase inicial de conexão, conhecida como *rendezvous*, na qual o usuário entra no ambiente, o modelo de comunicação é centralizado.

Um servidor chamado *registrar* é responsável em fazer a autenticação dos usuários. Uma vez que o usuário esteja conectado e autenticado, toda a comunicação ocorre diretamente entre os clientes, sem a intervenção do servidor *registrar*.

O Groupkit oferece alguns elementos de interface (*widgets*) que visam dar suporte à consciência de grupo (*awareness*), como: Teleponteiros, Visão Miniaturas, Visão Radar e Barra de rolagem multiusuário. As aplicações são desenvolvidas integrando esses elementos na aplicação compartilhada, sem a preocupação com aspectos de distribuição, como por exemplo, comunicação.

5.6. HABANERO

O Habanero [NCS2005] é uma plataforma de apoio à colaboração síncrona. Habanero tem como objetivo converter *applets* Java em objetos distribuídos. Isso não é feito automaticamente, é necessário alterar o código fonte do *applet*.

Qualquer *applet* existente pode ser convertido em uma aplicação compartilhada dentro do ambiente Habanero, através de pequenas alterações no código fonte. Este processo é chamado de “Habanerização” de um *applet* Java.

O ambiente colaborativo inclui um servidor de sessão e um cliente (Figura 5.6) que disponibiliza ao usuário uma variedade de aplicações (*Hablets*). A interface do cliente permite listar, criar, associar-se e interagir com sessões no servidor do Habanero [NCS2005].

O modelo de comunicação é centralizado. Baseado no modelo Cliente/Servidor (C/S). É necessário que o servidor esteja executando e disponível na rede para que as aplicações clientes possam se comunicar.

Os usuários podem criar um perfil (*user profile*) que pode conter informações pessoais, incluindo foto. Apesar disto a consciência de grupo (*awareness*), limita-se basicamente ao controle de sessão.

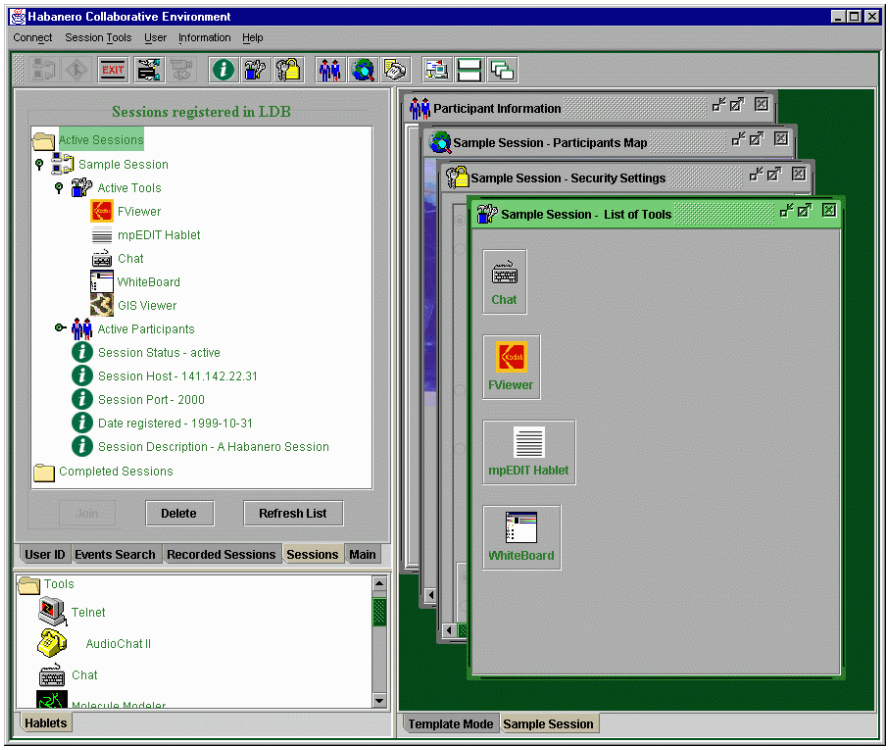


Figura 5.6 – Tela do cliente Habanero.

A idéia básica do Habanero é a distribuição (*broadcast*) das ações dos usuários para cada um dos participantes da sessão, o que caracteriza a arquitetura sincronizada, na qual a manutenção do estado compartilhado é alcançada sincronizando os ambientes através da replicação de eventos.

O Habanero oferece um conjunto de ferramentas, como por exemplo, quadro branco compartilhado, *chat* e uma ferramenta para visualização de estruturas de moléculas.

5.7. SHARE-KIT

Share-Kit [ROT1998] é uma plataforma para desenvolvimento de aplicações compartilhadas. Desenvolvido na linguagem C e baseado no ambiente Unix, Share-Kit utiliza o modelo de comunicação centralizado, na qual um servidor deve estar rodando e seu endereço na rede dever ser conhecido das aplicações clientes.

As aplicações clientes (Figura 5.7) são executadas diretamente a partir da linha de comando passando-se o endereço do servidor através de parâmetros. Share-Kit não possui gerenciamento de usuários e nem o conceito de sessão.

O foco do Share-kit é a aplicação compartilhada e não os usuários, não há comunicação direta entre usuários, somente o uso da aplicação compartilhada, essa era uma característica muito comum na primeira geração de sistemas de *groupware*.

O fato de não ter o conceito de sessão resulta em apenas um espaço virtual compartilhado, todos que executam a aplicação no mesmo momento estão no mesmo espaço virtual, e não tem como escolher com quem trabalhar ou trabalhar separadamente em outra atividade.

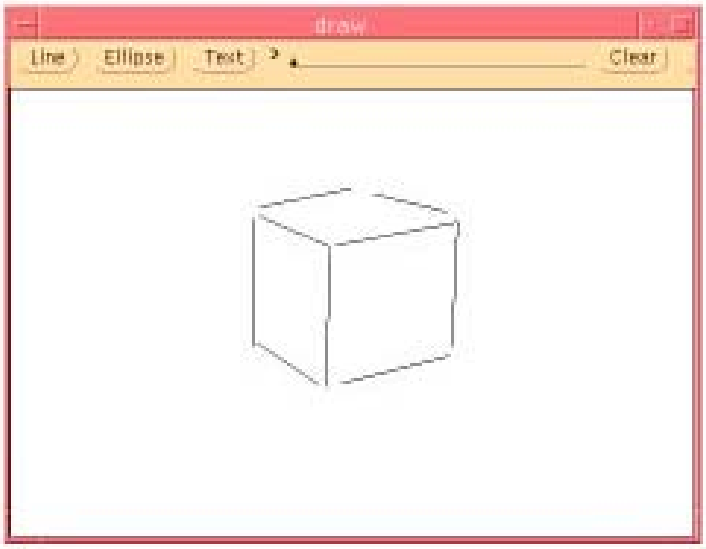


Figura 5.7 – Aplicação cliente do Share-Kit.

Share-Kit mantém o estado compartilhado das aplicações pela replicação de eventos através da transmissão simultânea para várias estações de trabalho de chamadas remotas de procedimentos (Multicast-RPC).

5.8. COMPARAÇÃO ENTRE COMPETIDORES

A Tabela 5.1 apresenta uma comparação entre os produtos competidores analisados. A maioria dos artigos que serviram de fonte de informação sobre os competidores não apresentaram detalhes de implementação

dos sistemas e nem se apresentam de forma homogênea. Logo, algumas informações só puderam ser inferidas a partir das características de cada competidor.

Tabela 5.1 – Comparação entre competidores

Ambientes / Critérios	ANTS	Dream Team	Groupkit	Habanero	NSTP	Rendezvous	Share-Kit
Modelo de comunicação	Centralizado	Descentralizado	Híbrido	Centralizado	Centralizado	Centralizado	Centralizado
Manutenção do estado compartilhado	Compartilhado	Sincronizado	Híbrida	Híbrida	Compartilhado	Compartilhado	Sincronizado
Consciência de colaboração	Sim	Sim	Sim	Não	Sim	Sim	Não
Controle de sessão	Sim	Sim	Sim	Sim	Sim	Sim	Não
Definição de papeis	Não	Não	Sim	Não	Sim	Sim	Não
Percepção	Sim	Sim	Sim	Sim	Sim	Sim	Não
Plataforma dependente	Não	Não	Sim	Não	Não	Sim	Sim

5.9. REQUISITOS LEVANTADOS A PARTIR DA ANÁLISE DE COMPETIDORES

A partir das informações coletadas na análise de competidores, pudemos identificar e levantar os requisitos da interação síncrona que estão incorporados nas plataformas de *groupware* síncronos encontrados e já identificados por estas plataformas. A seguir, apresentamos a lista de requisitos derivados da análise de competidores:

[RQ01] A plataforma de *groupware* deve fornecer serviço de comunicação entre as aplicações compartilhadas

Comunicação é o requisito básico dos sistemas colaborativos, sem a capacidade de comunicação não pode haver colaboração. Como pôde ser visto, as aplicações de *groupware* dependem essencialmente da comunicação para manter o controle do estado do ambiente distribuído.

No nível de aplicação, a comunicação assume seu papel funcional (comunicação entre processos) visando oferecer suporte para a comunicação entre as diversas instâncias da aplicação de grupo, tendo como principal objetivo a manutenção do ambiente virtual.

[RQ02] A plataforma de *groupware* deve fornecer serviço de comunicação entre os usuários das aplicações compartilhadas

A plataforma de grupo deve oferecer suporte à comunicação entre dois ou mais participantes humanos do ambiente compartilhado. Este tipo de comunicação pode incluir texto, voz, imagens, vídeos, gestos e outras formas de comunicação não verbal.

[RQ03] O projetista da plataforma de *groupware* deve escolher o modelo que será usado para fornecer comunicação entre as aplicações colaborativas

No modelo centralizado, à exemplo do modelo cliente/servidor clássico, adotado pela maioria dos competidores analisados. Neste modelo quando se incrementa o número de clientes participantes do ambiente virtual, o servidor torna-se o gargalo da comunicação, já que toda a comunicação entre os participantes é distribuída pelo servidor.

O modelo descentralizado conhecido como ponto-a-ponto, adotado pelo DreamTeam, pode superar esta limitação, mas em contrapartida aumenta a complexidade da aplicação. Neste modelo, os participantes trocam informações entre si, sem um servidor intermediário. Não há necessidade que todos os pontos estejam conectados diretamente entre si. No entanto, existe a necessidade de conexões indiretas e que todos os participantes devem ter conhecimento de quem está conectado ao ambiente virtual.

Algumas soluções intermediárias baseiam-se em modelos híbridos, derivados dos dois modelos anteriores, como é o caso do GroupKit que introduziu um Servidor, chamado Registrar, que fornece percepção de presença entre as aplicações clientes. Cada cliente toma ciência da presença dos demais através do Registrar, porém a comunicação entre os clientes é realizada diretamente.

[RQ04] A plataforma de *groupware* deve ser capaz de recuperar-se de falhas de comunicação (tolerância à falhas)

Sistemas de grupo são caracteristicamente sistemas distribuídos e conseqüentemente susceptíveis à falhas de comunicação. Isto implica que a plataforma deve ter mecanismos que permitam recuperar-se de atrasos da rede, falhas de comunicação, entre outros. Essa característica pode, por exemplo, garantir que um usuário que tenha uma pane em sua rede e perca a comunicação com o servidor, possa voltar a participar da atividade colaborativa, tão logo seja restabelecida a comunicação.

[RQ05] Sistemas colaborativos devem ser conscientes de sua colaboração

Sistemas colaborativos devem ser construídos com a intenção explícita de serem usados colaborativamente. Algumas plataformas para desenvolvimento de sistemas colaborativos, a exemplo do Habanero, podem compartilhar aplicações monousuárias, mas isso limita a interação entre os usuários porque as aplicações não foram construídas para serem colaborativas.

[RQ06] A plataforma de *groupware* deve fornecer um ambiente virtual compartilhado (*shared workspace*)

O Ambiente virtual deve ser compartilhado por todos os usuários do grupo e isso exige que qualquer ação realizada por um dos usuários seja refletida para todos os outros. Neste caso, a estrutura de comunicação é utilizada pela própria aplicação de grupo, sem a interferência do usuário, para manter a sincronia da visão compartilhada do ambiente.

[RQ07] A plataforma de *groupware* deve coordenar o acesso ao ambiente virtual compartilhado

O acesso concorrente necessita de políticas que resolvam conflitos de acesso ao mesmo objeto no ambiente compartilhado. O Rendezvous atacou o problema a partir de uma visão de escalonamento, baseada na literatura de sistemas operacionais. Porém Greenberg e Marwood [GRE1994] afirmam que não existe um esquema de controle de concorrência que atenda à todos os tipos de

aplicações de *groupware*, simplesmente porque o usuário é parte ativa do processo.

Políticas de acesso ao ambiente compartilhado são conhecidas na literatura como *Floor Policies* [ABD1997] [BOY1993] [GUE2001]. O *Floor* controla a habilidade do usuário de interagir com objetos compartilhados no ambiente. Alguns exemplos de políticas de acesso encontradas na literatura são:

Request-and-Get policy: Permite que o usuário receba o direito de interagir com objetos no ambiente compartilhado imediatamente após solicitar o *Floor*.

Request-and-Wait policy: Permite ao usuário requerer o *Floor*, mas apenas recebe direito de interagir quando o usuário que mantém o *Floor* liberá-lo.

No-Floor policy: Permite que qualquer participante interaja com o ambiente compartilhado. Ideal para aplicações simples, como *Whiteboard*.

[RQ08] A plataforma de *groupware* deve prover gerenciamento de sessão

Uma sessão em uma plataforma de *groupware* será necessariamente síncrona e tipicamente inicia-se com a entrada do primeiro usuário no ambiente compartilhado até a saída do último usuário do mesmo ambiente.

É necessário o gerenciamento das sessões dos usuários na plataforma, gerenciando grupos de trabalho e monitorando usuários que entram e saem dos grupos. Cada sessão representa um grupo de trabalho diferente. Usuários que trabalham em uma determinada sessão (grupo), não devem visualizar o trabalho de outros usuários que estejam em sessões diferentes, mesmo que estejam utilizando a mesma aplicação colaborativa.

[RQ09] A plataforma de *groupware* deve fornecer segurança de acesso a sessões

A plataforma de grupo deve fornecer um mecanismo de autenticação dos usuários, manter uma lista de sessões ativas e a lista de usuários associados a estas sessões. Opcionalmente podem ser criados perfis de usuários, que

definem quais aplicações o usuário pode executar e em que tipos de sessões, ou mesmo a que sessões, os mesmos podem associar-se.

[RQ10] A plataforma de *groupware* deve permitir a definição de papéis para os usuários que participam de uma atividade compartilhada

Muitas aplicações colaborativas exigem que os usuários desempenhem diferentes papéis durante a colaboração com diferentes direitos de acesso aos objetos no ambiente compartilhado. Certas operações como excluir podem ser restritas somente a certos tipos de usuários.

As aplicações educacionais, principalmente, necessitam diferenciar, por exemplo, entre professores, tutores e alunos, pois são usuários que certamente terão funções e privilégios diferenciados na atividade colaborativa.

[RQ11] A plataforma de groupware deve prover aos usuários informações de percepção úteis para a colaboração e coordenação da atividade colaborativa

Percepção é a compreensão que um indivíduo deve ter sobre as atividades dos outros, a qual irá prover um contexto para as suas próprias atividades. Esse contexto é utilizado para garantir que as contribuições individuais são relevantes para as atividades do grupo como um todo, e para avaliar as ações individuais com respeito a metas e progresso do grupo [DOU1992].

As informações de percepção são fornecidas aos usuários finais dos sistemas de *groupware* através de mecanismos de percepção. Os mecanismos de percepção correspondem, portanto, às técnicas utilizadas por um sistema para fornecer informações destinadas à percepção.

[RQ12] A plataforma de *groupware* pode prover *widgets* colaborativos que diminuam o esforço de desenvolvimento de novas aplicações colaborativas

Widgets são elementos de interface projetados para apoiar o desenvolvimento rápido de aplicações. Estes elementos empregam algum tipo de representação gráfica e são usualmente integrados à própria janela da aplicação colaborativa. O Groupkit [ROS1996], por exemplo, fornece uma coleção de

widgets de percepção como Teleponteiros, Visão Miniaturas, Visão Radar e Barra de rolagem multiusuário.

[RQ13] De preferência a plataforma de *groupware* deve multi-plataforma

Com a popularização dos sistemas operacionais livres e o grande avanço tecnológico de novos dispositivos, como *palms* e celulares, o ambiente de execução dos sistemas compartilhados tende a ser cada vez mais heterogêneo. Plataformas de grupo devem optar por linguagem multiplataforma que permitam a execução de suas aplicações em uma grande variedade de sistemas operacionais e dispositivos sem a necessidade de recompilação.

5.10. CONSIDERAÇÕES FINAIS

A análise dos produtos existentes na literatura permite o reaproveitamento de requisitos. A análise de competidores favoreceu o reaproveitamento de requisitos, que puderam ser aplicados no desenvolvimento dos protótipos que serão apresentados no próximo capítulo.

A análise de competidores é útil para: identificar as melhores práticas; levantar requisitos; avaliar os pontos fortes e fracos de nossos competidores; reutilizar experiência de design e colaborar com a especificação funcional de novos sistemas.

No próximo capítulo apresentamos os resultados da prototipação de uma plataforma de *groupware* (Plattus) e um componente síncrono de aprendizagem (Gérard).

6. PROTOTIPAÇÃO

6.1. PROTÓTIPO DA PLATAFORMA DE *GROUPWARE* PLATTUS

A plataforma de *groupware* visa oferecer suporte à construção de ferramentas de apoio a atividades colaborativas em ambientes virtuais de ensino na Web, fornecendo a infra-estrutura de comunicação e coordenação da atividade de grupo.

Com a plataforma é possível criar diferentes aplicações educacionais que poderão ser compartilhadas por um grupo de alunos, permitindo que o significado de conceitos abordados em sala de aula sejam progressivamente construídos mediante a ação colaborativa dos participantes de uma atividade sobre uma mesma interface compartilhada.

Nesta seção, apresentaremos os detalhes de implementação do protótipo da plataforma de *groupware*. No decorrer do texto quando uma referência do tipo **[RQxx]** aparecer, significa que um requisito levantado na análise de competidores foi atendido pelo protótipo.

6.1.1. ARQUITETURA DE DISTRIBUIÇÃO

Adotamos a arquitetura de distribuição híbrida, que apresenta características unificadas das arquiteturas replicada e distribuída. A manutenção do espaço compartilhado é baseada no componente modelo da arquitetura MVC, permitindo o acoplamento do tipo WYSIWIS relaxado, o que não obriga que todos os alunos estejam na mesma posição e vejam a mesma coisa. Este acoplamento acontece no nível do modelo, assim os usuários podem colaborar sobre o mesmo modelo, mas utilizando visões completamente diferentes.

6.1.2. COMUNICAÇÃO

A plataforma de *groupware* é composta de duas partes distintas: O Servidor de grupo, que é responsável pela comunicação e coordenação da atividade de grupo e a API de *groupware* que fornece todas as funcionalidades para conexão e comunicação entre as ferramentas colaborativas e o Servidor de grupo.

A API de *groupware* funciona como uma camada intermediária entre as ferramentas colaborativas e a rede, padronizando a comunicação e permitindo separar a preocupação com os objetivos específicos da aplicação da preocupação com os requisitos comuns aos programas colaborativos.

Toda a comunicação **[RQ01]** é baseada em mensagens que são enviadas entre as ferramentas colaborativas educacionais. A comunicação entre as aplicações colaborativas se dá de forma centralizada **[RQ03].** Todas as mensagens são enviadas pelos clientes para o servidor, o qual se encarrega de repassar para os demais usuários conectados. O protótipo suporta cinco tipos de mensagens:

Mensagens de Controle: mensagens usadas pela plataforma de *groupware* síncrona e que nunca chegam diretamente ao usuário. São mensagens trocadas somente entre o servidor de grupo e a API de suporte à ferramentas educacionais;

Mensagens de Erro: como o próprio nome já diz, mensagens de erro são usadas para reportar possíveis erros ocorridos no sistema;

Mensagens de Modelo: mensagens que encapsulam o modelo de dados do usuário. Cada alteração no modelo de dados requer que o modelo seja transmitido para todos a fim de atualizar suas visões;

Mensagens de *Chat* **[RQ02]**: utilizadas para que os usuários possam trocar mensagens de texto;

Mensagens de Aplicação: mensagens que podem ser usadas livremente pelo desenvolvedor da ferramenta educacional para trocar mensagens referentes ao domínio da aplicação.

6.1.3. CONSCIÊNCIA DE COLABORAÇÃO

A plataforma de *groupware* Plattus exige que a aplicação seja desenvolvida com a colaboração em mente **[RQ05]** e fornece uma API para prover os serviços de compartilhamento do ambiente. O que permite um melhor aproveitamento dos recursos compartilhados.

6.1.4. ESPAÇO COMPARTILHADO

No Plattus o modelo representa o ambiente virtual que deve ser compartilhado entre os usuários **[RQ06]**, a visão é apenas uma representação visual deste modelo e o controle é o responsável pela interação com o usuário e pela atualização do modelo.

Uma cópia da versão corrente do modelo fica armazenada no servidor bem como em cada máquina cliente, sendo o servidor responsável por atualizar cada cliente no momento em que o modelo é modificado. A cada modificação realizada no modelo pelo usuário que detém o controle, o cliente envia o modelo para o servidor, a fim de atualizar a cópia do mesmo no servidor, bem como para que o modelo possa ser enviado para todos os outros clientes.

6.1.5. CONTROLE DE SESSÃO

O controle de acesso implementado pelo protótipo é conhecido como *Request-and-Wait policy* **[RQ07]***. Neste modelo, se um* usuário deseja manipular objetos no ambiente compartilhado, deverá requisitar o controle do mesmo. O servidor irá gerenciar a fila de solicitações de controle, garantindo que cada usuário terá sua vez de modificar o modelo na ordem em que tais solicitações foram recebidas pelo servidor.

Para ter acesso ao ambiente compartilhado o usuário ao entrar no sistema deve informar um *login* e o nome do grupo ao qual deseja conectar-se. Porém, nenhuma verificação de senha ou qualquer autenticação é necessária para utilizar os serviços de colaboração oferecidos pelo protótipo.

O Plattus é responsável pelo gerencianmento dos grupos monitorando os usuários que entram e saem dos grupos. Cada grupo é associado a uma sessão de trabalho diferente **[RQ08]**. Usuários que trabalham em uma determinada sessão (grupo), não visualizam o trabalho de outros usuários que estejam em sessões diferentes, mesmo que estejam utilizando a mesma aplicação colaborativa.

6.1.6. DEFINIÇÃO DE PAPÉIS

O protótipo não fornece suporte para definição de papéis para os usuários que participam de uma atividade compartilhada. Esta funcionalidade deverá ser implementada nas próximas versões do sistema.

6.1.7. PERCEPÇÃO

A plataforma Plattus apresenta informações de percepção **[RQ11]** de sessão, indicando quando algum usuário entra ou sai do ambiente compartilhado. Esse é o tipo de percepção mais comumente encontrado nos ambientes de colaboração síncronos.

Além das informações de sessão, a plataforma fornece informações sobre qual usuário tem o controle do ambiente colaborativo e quais usuários requisitaram o mesmo.

6.1.8. COMPONENTES DE INTERFACES (WIDGETS)

Plattus fornece dois *widgets* que facilitam a criação de ferramentas colaborativas **[RQ12]**: A lista de usuários e o componente de *chat*.

A lista de usuários (Figura 6.1) é um componente que apresenta uma lista de usuários ativos no ambiente compartilhado. Além da lista de usuários, o componente apresenta informações de percepção, como quem tem o controle do ambiente compartilhado ([x]) e quais usuários requisitaram o controle ([r]).

Figura 6.1 – Componente "Lista de Usuários".

O outro componente de interface fornecido pelo Plattus é uma janela de *chat* (Figura 6.2), que permite aos usuários que participam de uma mesma sessão colaborativa trocarem mensagens de texto **[RQ02]**.

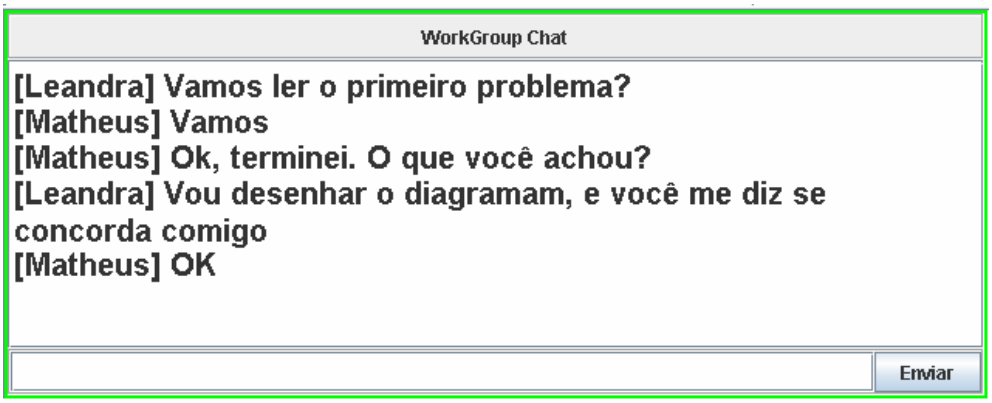


Figura 6.2 – Componente de *Chat*.

6.1.9. INDEPENDÊNCIA DE PLATAFORMA

Plattus foi desenvolvido com a linguagem Java, linguagem multiplataforma que permite a execução de suas aplicações em uma grande variedade de sistemas operacionais e dispositivos **[RQ13]**.

6.2. SERVIDOR PLATTUS

O servidor Plattus é o ponto central da plataforma de *groupware* Plattus. Ele é responsável pela comunicação e coordenação da atividade de grupo. Toda comunicação entre os clientes e o servidor é padronizada pela classe adaptadora chamada *WorkGroupConnection* que fornece uma API de *groupware* que fornece métodos de serviço para comunicação entre clientes e servidor.

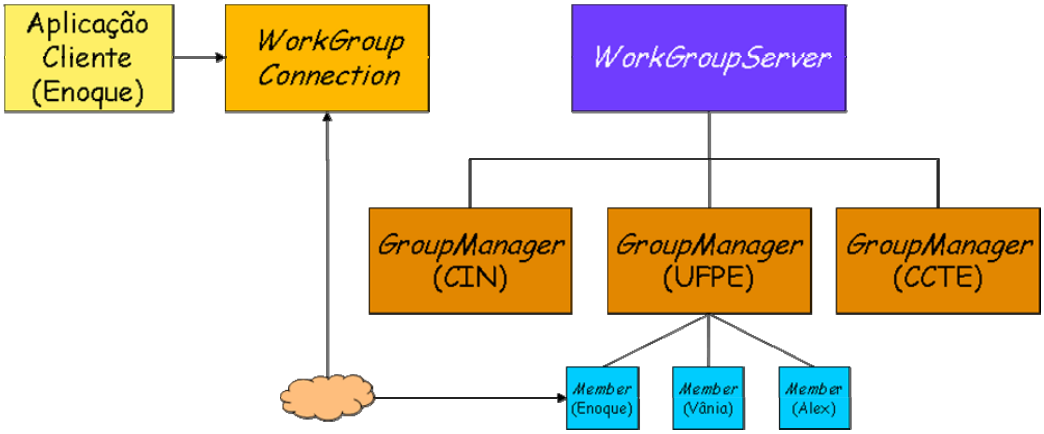


Figura 6.3 – Arquitetura simplificada de comunicação

Como pode ser visto no Figura 6.3, cada cliente se conecta ao servidor somente através da API fornecida pela classe *WorkGroupConnection*. Do lado servidor cada cliente é considerado membro de um grupo e para cada grupo o servidor cria um *GroupManager* (Gerenciador de grupo) que é responsável pela coordenação dos membros do grupo.

Toda comunicação entre clientes e servidor é feita através de mensagens. Um exemplo simples do modelo de comunicação pode ser visto na Figura 6.4 contendo três clientes. O gerenciador de grupos encarrega-se de distribuir as mensagens. O padrão de interação é o seguinte:

1. Durante a inicialização, todos os clientes conectam-se ao servidor através da Interface fornecida pela classe *WorkGroupConnection*. O gerenciador de grupo mantém o registro de todos os clientes subscritos no grupo.
2. O cliente que tem o controle chega ao ponto onde é necessário enviar uma mensagem. No caso do exemplo uma mensagem de modelo. Para isso, ele cria e publica a mensagem invocando o método *sendModelMessage*. A mensagem é colocada na fila.
3. Imediatamente, o Gerenciador de grupo envia a mensagem para os clientes através da chamada ao método *onMessage*.

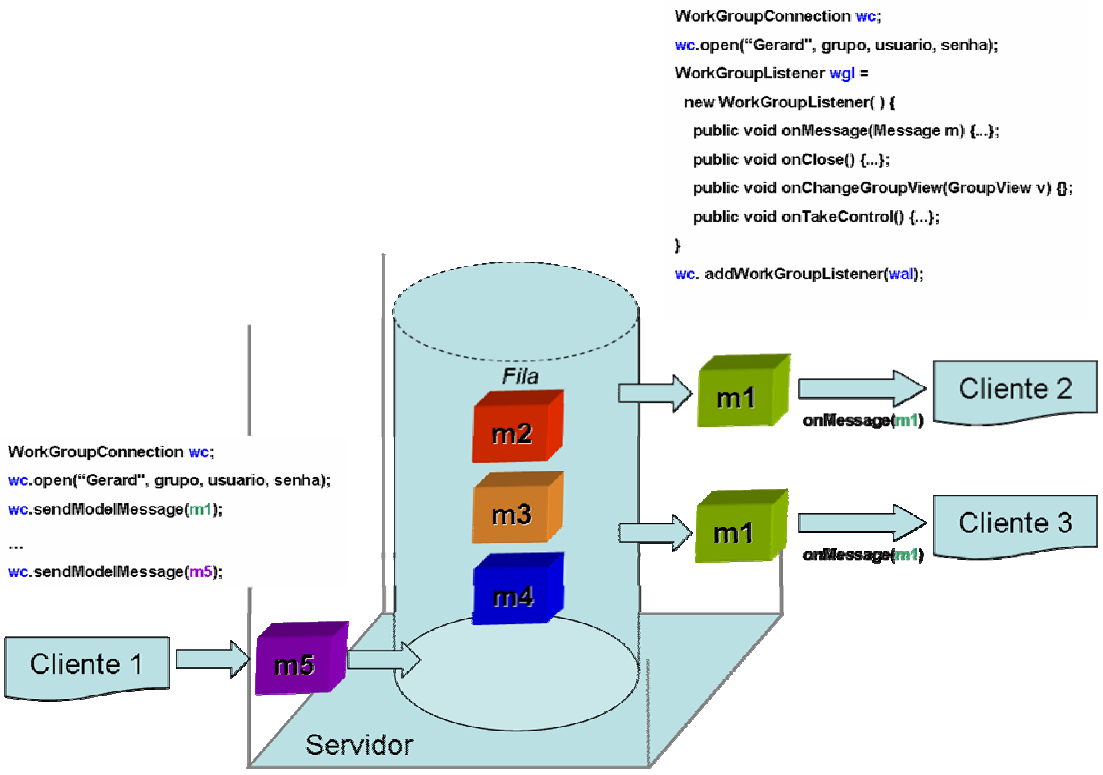


Figura 6.4 – Modelo de comunicação.

6.2.1. MENSAGENS

As Mensagens da plataforma Plattus têm um formato básico simples mas altamente flexível, que permite a trocar mensagens com aplicações em plataformas heterogêneas.

Uma mensagem da plataforma Plattus é composta de três partes distintas:

- Cabeçalhos
- Data de recebimento (timestamp)
- Corpo (opcional)

Cabeçalhos

O cabeçalho da mensagem pode conter vários campos para valores que clientes podem utilizar para identificar mensagens. Os clientes estão livres

para adicionar ler campos ao cabeçalho das mensagens, porém a função que terá cada cabeçalho adicionado pelos clientes será definida e controlada pelas próprias aplicações clientes.

No futuro campos predefinidos pela plataforma serão adicionados às mensagens para controle da comunicação, como por exemplo, um identificador único seqüencial que permitirá a ordenação das mensagens.

Data de recebimento

A data de recebimento da mensagem armazena a informação de data/hora em que a mensagem foi recebida pelo servidor de *groupware*. O motivo pelo qual é a data/hora do servidor é para evitar problemas com a diferença de data entre os diversos clientes envolvidos na comunicação.

Corpo da Mensagem

O corpo da mensagem contém os dados que realmente estão sendo transmitidos. A plataforma Plattus define 5 tipos diferentes de mensagens:

6.2.2. API DE COMUNICAÇÃO CLIENTE/SERVIDOR

Como já foi dito antes toda a comunicação entre cliente e servidor é definida pela API de *groupware* encapsulada na classe *WorkGroupConnection*. A seguir apresentamos os principais métodos definidos pela API:

Open – O método Open é responsável por estabelecer uma conexão entre cliente e servidor. O método recebe como parâmetros o a identificação da aplicação, o nome do grupo o login e a senha do usuário.

SendMessage – Existe um método para o envio de cada tipo de mensagem suportada pela plataforma: sendModelMessage, sendChatMessage, sendApplicationMessage. Mensagens de controle e de erro são utilizadas internamente pela plataforma e não estão disponíveis métodos para envio destes tipos de mensagens pelos clientes.

RequestFloorControl – Utilizado para solicitar o controle do modelo da aplicação.

LeaveFloorControl – Utilizado para abandonar o controle do modelo da aplicação.

Close – Fecha a conexão estabelecida com o servidor de *groupware*.

isOpenned – Retorna verdadeiro se a conexão com o servidor estiver aberta.

addWorkGroupListener – Adiciona um *Listener* para receber mensagem e eventos de forma assíncrona, ou seja, sem bloqueio do cliente.

6.2.3. EXEMPLO DE ENVIO E RECEBIMENTOS DE MENSAGENS

Dentro da arquitetura Plattus são necessários alguns passos para estabelecer uma conexão e preparar para enviar e receber mensagens:

1. Obter uma *WorkGroupConnection* – Qualquer cliente precisam obter uma *WorkGroupConnection*, que será usada para estabelecer uma conexão.

```
WorkGroupConnection connection;
connection = new WorkGroupConnection(endereco_servidor);
```

2. Abrir uma conexão – Após obter uma *WorkGroupConnection* a aplicação cliente pode usa-la para abrir uma conexão com o Servidor Plattus.

```
connection.open("Gerard", grupo, usuario, senha);
```

3. Envio de mensagens – Como vimos existem 5 tipos diferentes de mensagens na arquitetura Plattus. O cliente utiliza o método correspondente para enviar a mensagem correta:

```
connection.sendModelMessage(modelo);
```

4. Recebimento de mensagens – Para receber mensagens e eventos do servidor de grupo o cliente deve implementar a interface *WorkGroupListener*.

   Apresentamos abaixo a interface *WorkGroupListener* que define os métodos que devem ser implementado pelo *Listener*.

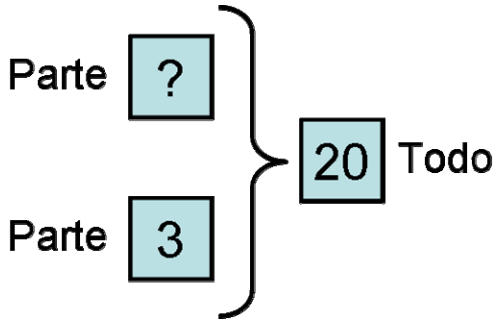
```
public interface WorkGroupListener  {
   public void onMessage(Message message);
   public void onClose();
   public void onChangeGroupView(GroupView view);
   public void onTakeControl();
}
```

6.3. COMPONENTE COLABORATIVO GÉRARD

Para validar a plataforma Plattus e a arquitetura de colaboração, foi desenvolvido um componente educacional, chamado Gérard, que permite aos alunos interagirem na resolução de problemas básicos no campo conceitual das estruturas aditivas, manipulando diagramas padrões, definidos por Vergnaud [VER1997], como representação dos problemas propostos. O nome Gérard é uma homenagem ao pesquisador Gérard Vergnaud, propositor da Teoria dos Campos Conceituais [VER1997].

6.3.1. ESTRUTURAS ADITIVAS

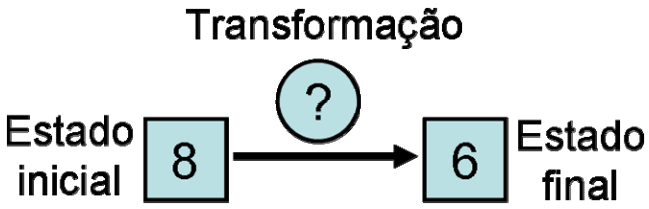
Gérard Vergnaud, propositor da Teoria dos Campos Conceituais, em seus estudos sobre as estruturas aditivas, fez uma classificação de problemas pelas dificuldades dos problemas e pelos raciocínios requeridos para resolvê-los. Vergnaud [VER1997] considera o campo conceitual das estruturas aditivas formado por três grupos básicos de problemas, classificados segundo suas características em: composição, transformação e comparação.



Problema: Vânia e Gabriel colecionam chaveiros. Vânia tem 8 chaveiros e Gabriel tem 7. Quantos chaveiros eles tem?

Figura 6.5 – Exemplo de diagrama e problema de composição.

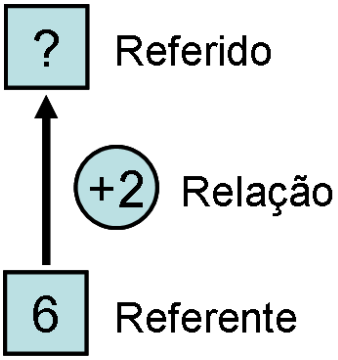
Problemas de composição, apresentado na Figura 6.5, representam situações que envolvem parte-todo. Onde o aluno, ao resolver, deve juntar uma parte com outra para obter o todo, ou subtrair uma parte do todo para obter a outra parte.



Problema: João tinha 8 carrinhos, ganhou mais 6 de sua mãe. Quantos carrinhos João tem agora?

Figura 6.6 – Exemplo de diagrama e problema de transformação.

A classe de problemas de transformação (Figura 6.6) trata de situações onde existe uma idéia temporal envolvida. Existe um estado inicial que sofre uma transformação de perda ou ganho, chegando ao estado final com uma nova quantidade.



Problema: Pedro tem 6 carrinhos, João tem dois a mais que Pedro. Quantos carrinhos tem João?

Figura 6.7 – Exemplo de diagrama e problema de comparação.

Os problemas de comparação, representados na Figura 6.7, implicam na comparação de duas quantidades: uma denominada referente e a outra referido. No exemplo, a quantidade de carrinhos de João é apontada em relação a quantidade de carrinhos de Pedro. A quantidade de carrinhos de Pedro é a referência (referente), pois para obter-se a quantidade de carrinhos de João (referido) é necessário saber a quantidade possuída por Pedro.

Dentro destes três tipos básicos existe ainda a possibilidade de gerar outras variações de problemas diferentes dos problemas apresentados nos exemplos anteriores. Como pode ser visto na Figura 6.8, existem diversas variações para os problemas e diagramas básicos das estruturas aditivas. Como por exemplo, tomemos o diagrama de composição (à esquerda), um possível problema para este diagrama seria: Ao redor de uma mesa estão sentadas 20 crianças, destas, 3 são meninas. Quantos meninos estão sentados ao redor da mesa?

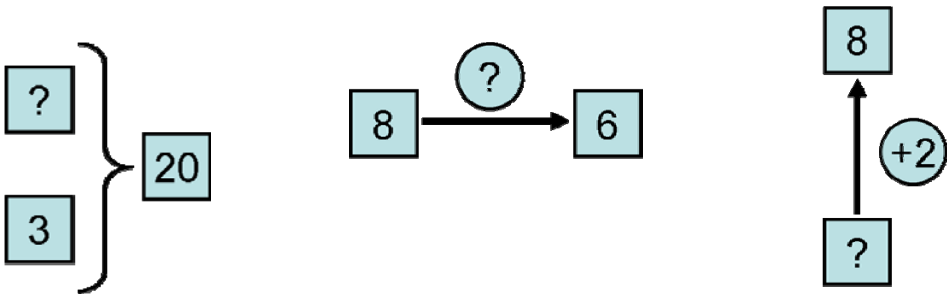
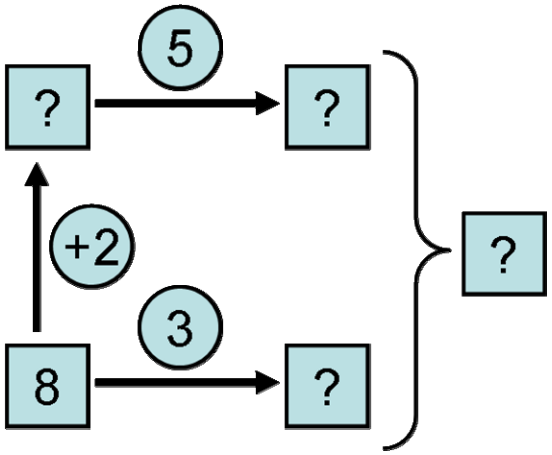


Figura 6.8 – Exemplo de variações dos diagramas básicos.

Ainda é possível a existência de situações problema que compõem duas ou mais estruturas. Como pode ser visto na Figura 6.9, existe uma infinidade de combinações entre as diversas estruturas, que podem gerar problemas mais complexos.



Problema: Carlos e seu irmão André colecionam selos. Carlos tinha 8 selos e André tinha 2 selos a mais que seu irmão. Seu pai chegou de viagem e deu 3 selos para Carlos e 5 selos à André. Quantos selos Carlos e André têm ao todo?

Figura 6.9 – Estrutura aditiva composta.

6.3.2. PROTÓTIPO DO COMPONENTE GÉRARD

O Gérard é um software que tem o objetivo de favorecer a aprendizagem colaborativa no campo conceitual das estruturas aditivas, através da manipulação direta dos diagramas de Vergnaud, para resolução conjunta de situações-problemas.

A interface do Gérard (Figura 6.10) é constituída de três áreas principais: uma lista de usuários ativos ❶, fornecida pela plataforma Plattus; um *chat* para a comunicação dos usuários ❷, também fornecido pela plataforma e uma área de trabalho compartilhada ❸, utilizada para o desenho dos diagramas.

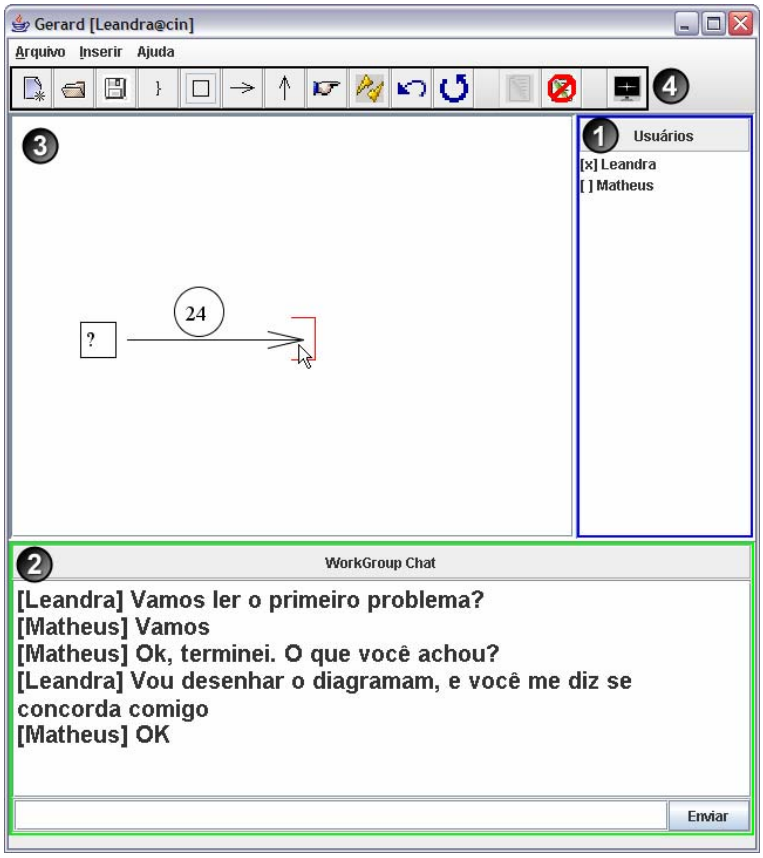


Figura 6.10 – Interface do ambiente Gérard.

A interface do protótipo Gérard também apresenta uma barra de ferramentas ❹, que fornece atalhos para as opções contidas nos menus da aplicação. A barra de ferramentas do Gérard é composta dos seguintes botões:

Tabela 6.1 – Botões da barra de ferramentas do Gérard

	Novo: Limpa a área compartilhada e inicia um novo diagrama.
	Abrir: Abre um diagrama previamente salvo no sistema local de arquivo do usuário.
	Salvar: Salva o diagrama no sistema local de arquivo do usuário.
	Composição: Insere um diagrama básico de composição.
	Valor: Insere no diagrama um valor absoluto.

	Transformação: Insere um diagrama básico de transformação.
	Comparação: Insere um diagrama básico de comparação.
	Seleção: Seleciona um objeto no diagrama.
	Solução: Resolve o diagrama.
	Desfazer: desfaz a última ação realizada no espaço compartilhado.
	Refazer: refaz a última ação desfeita.
	Solicitar controle: Solicitar o direito de editar o diagrama.
	Abandonar controle: Abandonar o direito de editar o diagrama.
	Sair: Sair do Gérard.

No Gérard, somente um usuário detém o controle da aplicação em um determinado momento, ou seja, apenas aquele que tem o controle consegue criar/modificar a representação do problema.

Enquanto isso, os demais usuários só conseguem assistir a interação e colaborar através do *chat* disponível no ambiente. Qualquer usuário pode solicitar o controle da aplicação, mas este só será passado quando o usuário que o detém abandoná-lo voluntariamente. Se mais de um usuário requisitar o controle da aplicação, este será passado na mesma ordem em que foi solicitado. É possível identificar quem detém ou requisitou o controle da aplicação através do sinal [r] localizado ao lado dos nomes dos usuários ativos.

Uma característica importante do Gérard é a verificação da consistência do diagrama a cada modificação da estrutura. Essa permite desfazer/refazer qualquer modificação realizada. Inconsistências são geradas quando valores representando termos ou o próprio resultado do problema estão incorretos. Para destacá-las o sistema utiliza o recurso de cores para dar *feedback* ao usuário, por exemplo, qualquer erro inserido em um diagrama é destacado em vermelho.

6.4. CONSIDERAÇÕES FINAIS

O desenvolvimento de protótipos foi útil em dois aspectos. O primeiro refere-se ao entendimento dos problemas relacionados ao desenvolvimento de sistemas colaborativos e a exploração dos aspectos críticos dos requisitos. Somente quando se tenta implementar um sistema deste tipo, por mais simples que seja, é que nos deparamos com a complexidade envolvida em conceitos aparentemente muito simples como *floor control* e definição de papéis.

Um outro aspecto do uso de protótipo é ter um produto concreto que pode ser utilizado para negociar com o usuário o entendimento do sistema e o levantamento de novos requisitos.

Como poderá ser visto no próximo capítulo, o protótipo do componente Gérard foi útil para realização de testes com usuários que geraram o cenário apresentado na primeira iteração e ajudou a encontrar novos requisitos para a plataforma de *groupware*.

7. ANÁLISE DE CENÁRIOS

7.1. PRIMEIRA ITERAÇÃO

A história contada neste cenário é fictícia, mas foi baseada em trabalho de campo realizado no trabalho de mestrado de Vânia Alves [ALV2005], no qual a autora observou vários usuários usando a ferramenta Gérard. A primeira iteração reflete uma série de situações reais experimentadas por vários usuários no uso da ferramenta Gérard.

O cenário descreve o uso da ferramenta Gérard por dois alunos hipotéticos de pedagogia, Matheus e Leandra, que procuram o professor para tirar dúvidas sobre a aula de estruturas aditivas. A tarefa proposta pelo professor prevê a resolução de várias situações problemas, mas somente a interação dos alunos para solução do primeiro problema é apresentada.

7.1.1. ATORES

Os Alunos Leandra e Matheus e o professor.

7.1.2. CONTEXTO

Dois alunos de pedagogia, Matheus e Leandra, procuram o professor no final da aula sobre estruturas aditivas e lhe apresentam algumas dificuldades na compreensão dos diagramas de Vergnaud [VER1997]. O Professor propõe então aos alunos uma sessão de utilização de um protótipo de uma ferramenta colaborativa, chamada Gérard, como atividade docente para tentar elucidar alguns conceitos através da manipulação direta dos diagramas no computador. Os alunos mostram-se inseguros para realização da atividade, pois alegam nunca terem utilizado a referida ferramenta. O professor os acalma informando que prepararia todo o ambiente para a realização da atividade e que realizaria uma sessão inicial para ensiná-los a utilizar a ferramenta.

7.1.3. AMBIENTE (SETTING)

Matheus e Leandra estão em um laboratório que tem 12 computadores dispostos em uma grande bancada que atravessa a sala em quase toda sua extensão. Os computadores na bancada estão voltados em direções opostas

permitindo um melhor aproveitamento do espaço do ambiente. Todos os computadores estão ligados a uma rede local, que permite também acesso à Internet.

No canto esquerdo de quem entra no laboratório encontra-se uma impressora (Figura 7.1-A) que é compartilhada por todos através da rede local. Na parede lateral direita de quem entra vê-se um grande quadro branco (Figura 7.1-B), que é utilizado pelos professores durante as aulas práticas no laboratório. Matheus sentou-se ao computador no fundo do laboratório (Figura 7.1-C), enquanto que Leandra preferiu o computador mais próximo à porta de entrada (Figura 7.1-D), que fica do lado simetricamente oposto ao computador escolhido por Matheus. No outro computador ao fundo do laboratório (Figura 7.1-E) foi instalado o Servidor que controla a colaboração entre os alunos.

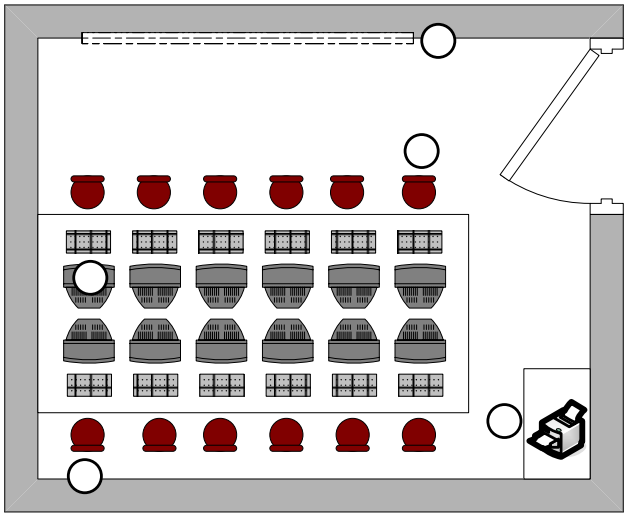


Figura 7.1 – Laboratório de informática.

O professor prepara todo o ambiente para realização da tarefa, prepara e testa o sistema nos computadores, também instala e configura o servidor de grupo. O professor também transcreve no quadro a tarefa que deveria ser executada pelos alunos, juntamente com as regras que ambos deveriam seguir durante a realização da atividade. Antes do início da atividade, o professor apresenta aos alunos instruções de como utilizar o sistema. Durante a realização

da atividade espera-se que os alunos alternem-se no controle da ferramenta para modelagem dos diagramas e que se comuniquem e negociem durante a realização da tarefa através do *chat* da ferramenta.

As regras definidas pelo professor para realização da atividade, transcritas por ele no quadro, são:

a) Vocês devem criar colaborativamente os diagramas que representam os problemas de adição e/ou subtração propostos;

b) Vocês devem ajudar-se mutuamente e interagir através do *chat* disponível no sistema, a fim de modelar tais problemas;

c) Não será permitido comunicar-se verbalmente e toda comunicação deverá ser efetuada através da ferramenta;

d) Vocês estão livres para negociar como deve se dar o processo de resolução;

e) Sempre que seu parceiro solicitar o controle da ferramenta e você não concordar com isso, vocês devem negociar via *chat* e a atividade só poderá ser retomada quando houver algum consenso;

f) Para que uma tarefa seja considerada completa, vocês devem estar em comum acordo e devem deixar isso bem claro concordando mutuamente através do *chat*.

Os problemas propostos pelo professor são os seguintes:

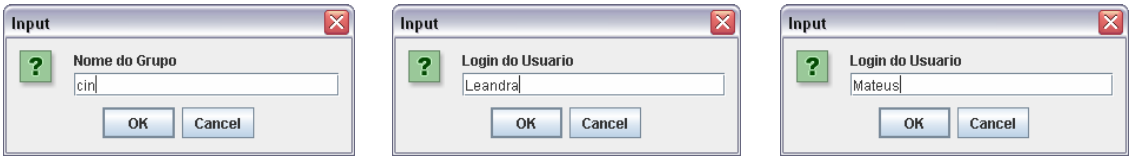
1) Joana tinha algumas revistas. Seu tio chegou de viagem e trouxe-lhe de presente 24 revistas para sua coleção. Joana tem agora 78 revistas. Quantas ela tinha antes?

2) Vera e Solange são feirantes. Vera vendeu 36 laranjas e vendeu 18 a mais que Solange. Quantas laranjas Solange vendeu?

7.1.4. ROTEIRO (PLOT)

Antes do início da atividade, o professor apresentou aos alunos instruções de como utilizar o sistema. Ambos acharam de fácil assimilação, pois já

estavam acostumados a usar outras ferramentas com funcionalidade equivalentes. Tanto Leandra como Matheus estavam ansiosos para utilizar a ferramenta.



(a) (b) (c)

Figura 7.2 – Telas de escolha do grupo e *login*.

Matheus e Leandra iniciam a atividade executando a ferramenta através de um atalho na área de trabalho. Inicialmente, eles escolhem o grupo ao qual conectar. Leandra escolhe o grupo “cin”, conforme sugerido pelo professor, Matheus não lembra muito bem da instrução do professor e tem dúvidas sobre o nome do grupo, mas após alguns segundos lembra que o nome do grupo acordado é “cin” (Figura 7.2a). Imediatamente após a escolha do grupo eles escolhem um apelido (*login*) com o qual serão referenciados no ambiente virtual. Leandra escolheu seu próprio nome como *login* (Figura 7.2b) e seguindo a mesma lógica, Matheus também preferiu seu nome original (Figura 7.2c). Após efetuarem o *login* no ambiente eles puderam visualizar a tela principal da ferramenta.

Leandra **Matheus**

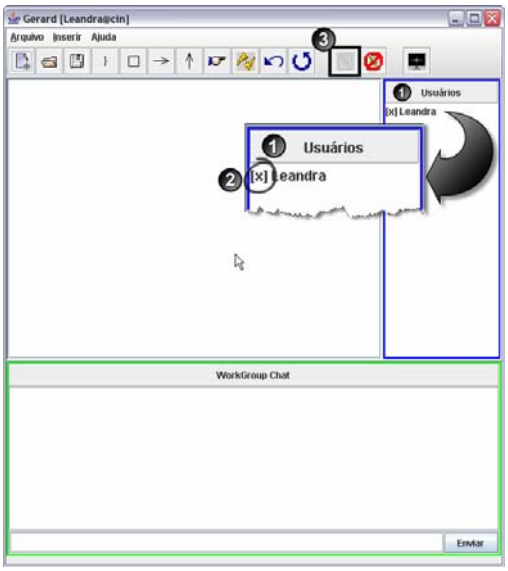


Figura 7.3 – Leandra entra no ambiente

Na Figura 7.3: Leandra é a primeira a entrar no ambiente. Ela percebe isto, pois é a única que consta na lista de usuários❶. Ela leva algum tempo para lembrar que o indicador [x] na frente de seu nome representa que ela tem o controle da ferramenta❷. Então, ela percebe que todos os botões da barra de ferramentas estão habilitados, com exceção de um, que serve para solicitar o controle da ferramenta❸.

Leandra **Matheus**

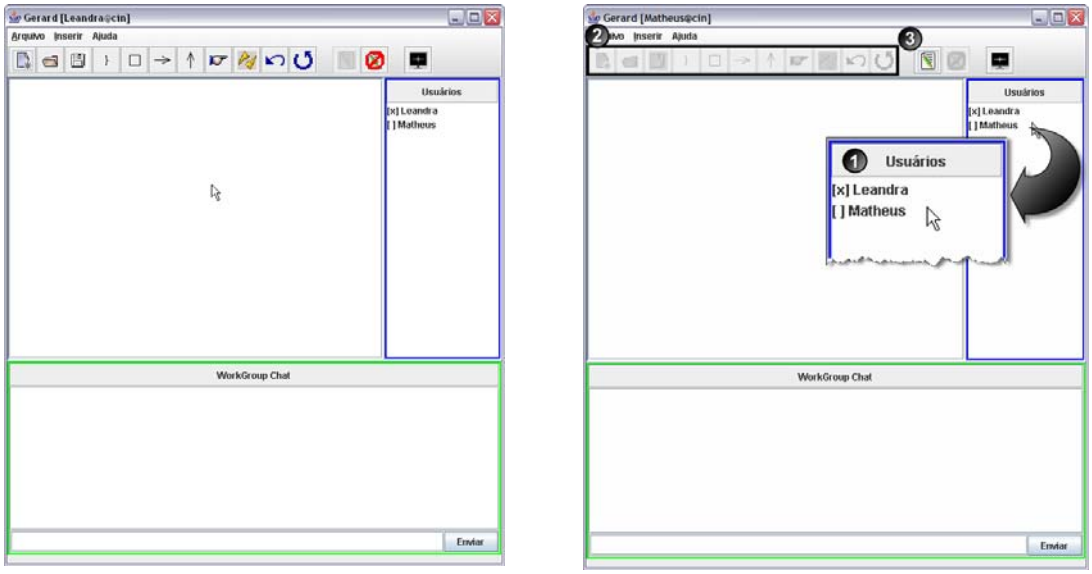


Figura 7.4 – Matheus entra no ambiente

Na Figura 7.4: Matheus entra logo em seguida, ele vê através da lista de usuários que Leandra já se encontra no ambiente❶. Ele queria ter o controle da ferramenta e lamenta o fato de ter demorado em lembrar o nome do grupo, quando lembra que o professor explicou que o primeiro a entrar no ambiente recebe automaticamente controle. Então ele percebe que quase todos os botões estão desabilitados❷, com exceção de um que serve para solicitar o controle da ferramenta❸.

Leandra **Matheus**

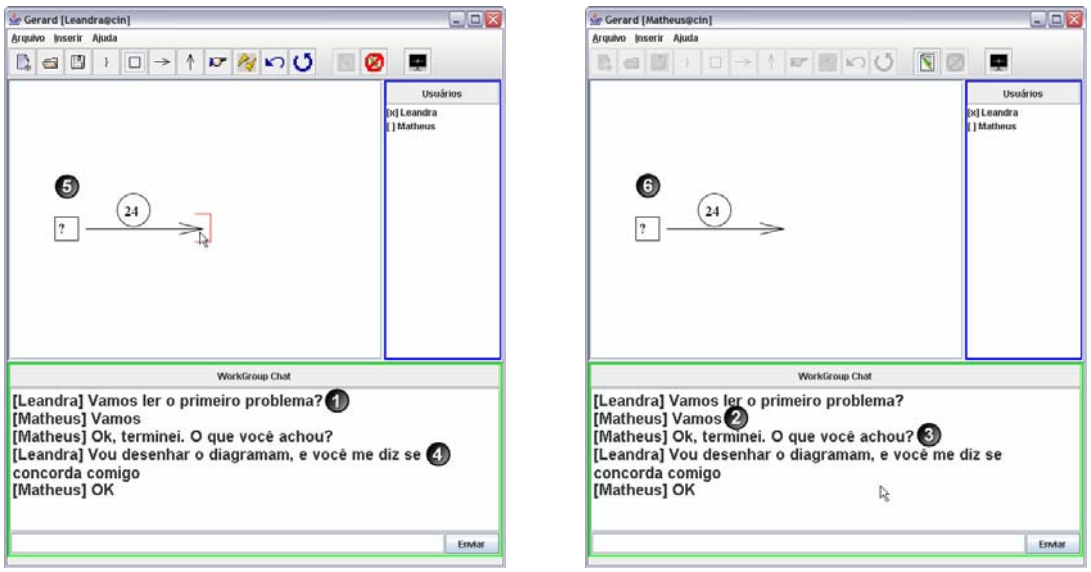


Figura 7.5 – Leandra desenha o diagrama no ambiente.

Na Figura 7.5: Leandra resolve convidar Matheus para ler o primeiro problema, lembra da regra (c) e convida o colega através do *chat*❶. Ele concorda❷ e após alguns minutos, que cada um reservou para ler atentamente o problema, Matheus indaga Leandra sobre o que ela achou da questão❸. Leandra acha que melhor do que expressar-se através do *chat* seria desenhar o diagrama❹. Matheus demora-se em responder, pois queria discutir pelo *chat*, mas concorda. Então Leandra começa a desenhar❺.

Matheus assiste sua colega desenhar o diagrama que representa a situação-problema, no ambiente compartilhado❻. Enquanto ela desenha, ele distrai-se clicando nos menus e procurando conhecer mais sobre a ferramenta. Mas como ele nada pode fazer, volta a observar o que a colega está fazendo.

Leandra **Matheus**

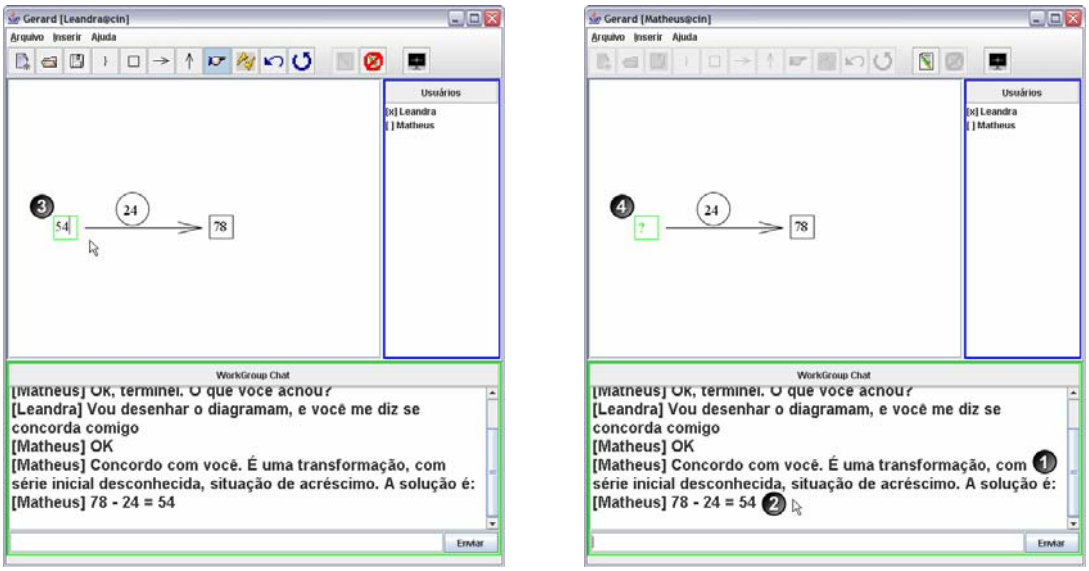


Figura 7.6 – Leandra termina de desenhar o diagrama.

Na Figura 7.6: Leandra termina de desenhar o diagrama e Matheus ao vê que a colega terminou de desenhar faz comentários através do *chat*, tentando mostrar que tem domínio do assunto❶. Ele também infere a solução do problema e apresenta para ela através do *chat*❷. Leandra pega o resultado oferecido pelo colega e completa o diagrama❸ para demonstrar que concorda com a resposta dele. Matheus vê a colega editar o diagrama para adicionar a resposta sugerida por ele❹.

Leandra **Matheus**

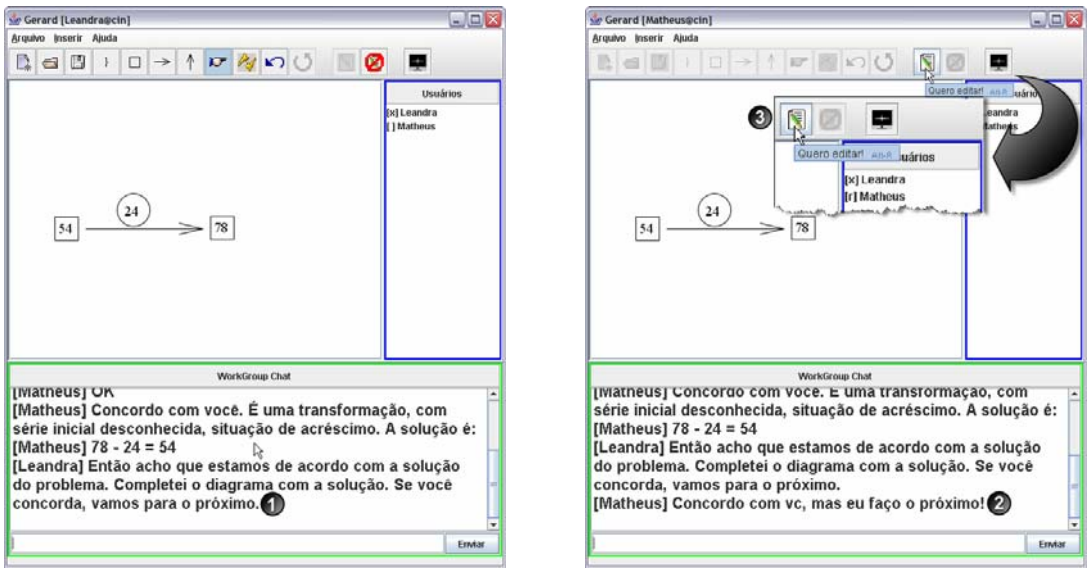


Figura 7.7 – Leandra e Matheus concluem a primeira tarefa.

Na Figura 7.7: Para completar a tarefa Leandra deixa explícito no *chat* que concorda com o colega e o convida para passar para o próximo problema❶. Matheus concorda com sua colega e expressa isso claramente através do *chat*, mas ressalta que pretende ter o controle da ferramenta para resolver o próximo problema❷. Então, ele solicita o controle através da ferramenta❸.

Leandra **Matheus**

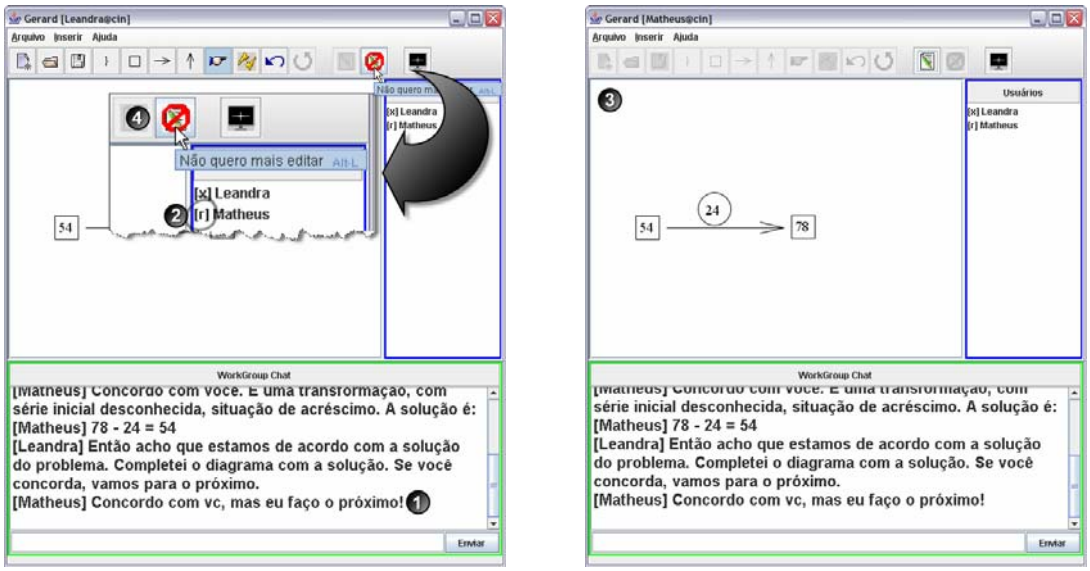


Figura 7.8 – Leandra passa o controle para Matheus.

Na Figura 7.8: Leandra recebe a mensagem do seu colega❶ e só neste momento percebe na lista de usuário o indicador [r] na frente do nome de Matheus❷, isso indica que o colega solicitou o controle da ferramenta. Matheus aguarda que Leandra lhe passe o controle❸. Leandra então concorda em passar o controle da ferramenta para ele e pressiona o botão❹ que realiza esta ação.

Leandra **Matheus**

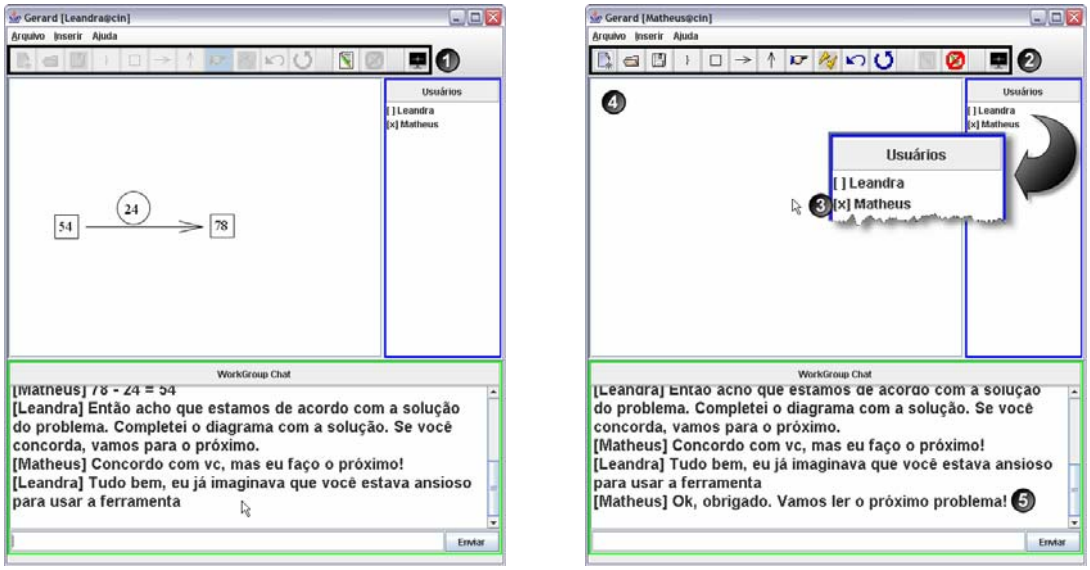


Figura 7.9 – Matheus recebe o controle do ambiente.

Na Figura 7.9: Após passar o controle, Leandra percebe que os botões da sua barra de ferramenta ficam quase todos desabilitados❶. Matheus recebe o controle e observa que os botões da sua barra de ferramentas ficaram habilitados❷ e percebe na lista de usuários o indicador [x] na frente de seu nome❸, que representa que ele tem o controle da ferramenta. Ele limpa a área compartilhada❹ para remover o diagrama do problema anterior e convida sua colega para ler o próximo problema❺.

7.1.5. ROTEIRO POSITIVO

Matheus e Leandra iniciam a atividade executando a ferramenta através de um atalho na área de trabalho. Inicialmente eles escolhem o grupo ao qual conectar. Eles sabem que devem escolher o mesmo grupo e previamente tinham

acordado que se conectariam ao grupo “cin”. Imediatamente após a escolha do grupo eles escolhem um apelido (*login*) com o qual serão referenciados no ambiente virtual. Leandra escolheu seu próprio nome como *login* e seguindo a mesma lógica, Matheus também preferiu seu nome original. Após efetuarem o *login* no ambiente eles puderam visualizar a tela principal da ferramenta.

Leandra é a primeira a entrar no ambiente. E percebe isto, pois é a única que aparece na lista de usuários. Ela lembra da instrução do professor e reconhece que o indicador [x] na frente de seu nome representa que ela tem o controle da ferramenta. Ela também percebe que todos os botões da barra de ferramentas estão habilitados, com exceção de um, que serve para solicitar o controle da ferramenta.

Matheus entra logo em seguida, e vê através da lista de usuários que Leandra já se encontra no ambiente. Ele já esperava que Leandra tivesse o controle da ferramenta, pois lembra que o professor explicou que o primeiro a entrar no ambiente recebe automaticamente controle. Ele também não estranha o fato de quase todos os botões da ferramenta estarem desabilitados, porque sabe que somente quem tem o controle da ferramenta é que pode manipular a área de trabalho compartilhada.

Leandra resolve convidar Matheus para ler o primeiro problema, mas lembra da regra (c) e convida o colega através do *chat*. Ele concorda e após alguns minutos, que cada um reservou para ler atentamente o problema, Matheus indaga Leandra sobre o que ela achou da questão. Leandra acha que melhor do que se expressar através do *chat* seria desenhar o diagrama. Matheus concorda prontamente. Então Leandra começa a desenhar.

Matheus assiste sua colega desenhar o diagrama, que representa a situação-problema, no ambiente compartilhado. Enquanto ela desenha, ele concorda com o diagrama e vai acompanhando o desenrolar da tarefa concordando com cada passo realizado por sua colega.

Leandra termina de desenhar o diagrama e Matheus ao vê que a colega terminou de desenhar faz comentários através do *chat* para que ela perceba que

ele está totalmente de acordo. Ele também infere a solução do problema e apresenta para ela através do *chat*. Leandra pega o resultado oferecido pelo colega e completa o diagrama, para demonstrar que concorda com a resposta dele. Matheus percebe que Leandra concorda quando a vê editar o diagrama para colocar a resposta sugerida por ele.

Para completar a tarefa, Leandra deixa explícito no *chat* que concorda com o colega e o convida para passar para o próximo problema. Matheus concorda com sua colega e expressa isso claramente através do *chat*. Mas ressalta que pretende ter o controle da ferramenta para resolver o próximo problema. Então, ele solicita o controle através da ferramenta.

Quando Leandra recebe a mensagem do seu colega, ela já havia percebido na lista de usuário o indicador [r] na frente do nome de Matheus, que indica que o colega solicitou o controle da ferramenta. Leandra então prontamente concorda em passar o controle da ferramenta para ele e pressiona o botão que realiza esta ação.

Após passar o controle Leandra percebe que os botões da sua barra de ferramenta ficam quase todos desabilitados. Matheus recebe o controle e observa que os botões da sua barra de ferramentas ficaram habilitados e percebe na lista de usuários o indicador [x] na frente de seu nome, o qual representa que ele tem o controle da ferramenta. Ele limpa a área compartilhada para remover o diagrama do problema anterior e convida sua colega para ler o próximo problema.

7.1.6. ROTEIRO NEGATIVO

Os requisitos aparecem mais facilmente no roteiro negativo, pois é o que requer uma visão crítica sobre a interação dos usuários com o sistema. No decorrer do texto quando uma referência do tipo **[RQ01]** aparecer, significa que um requisito foi levantado. Todos os requisitos levantados nesta seção serão apresentados em uma tabela na próxima seção.

Matheus e Leandra iniciam a atividade executando a ferramenta através de um atalho na área de trabalho. Inicialmente, eles escolhem o grupo ao qual conectar. Leandra escolhe o grupo “cin”, conforme sugerido pelo professor, mas

Matheus não lembra muito bem da instrução do professor e escolhe como nome do grupo “ufpe” **[RQ14]**. Ele gostaria de saber se Leandra já está neste grupo, para ter certeza de que está fazendo a escolha certa **[RQ15]**. Imediatamente após a escolha do grupo, eles escolhem um apelido (*login*) com o qual serão referenciados no ambiente virtual. Leandra escolheu seu próprio nome como *login* e seguindo a mesma lógica, Matheus também preferiu seu nome original **[RQ16]**.

Matheus questiona o professor sobre a necessidade de uso de senhas para evitar que outros usem seu *login*. O professor informa que a versão que eles estão usando é apenas um protótipo e que o normal seria o aluno autenticar-se no ambiente com seu *login* e senha cadastrados previamente **[RQ17]**.

Após efetuarem o *login* no ambiente eles visualizam a tela principal da ferramenta. Leandra é a primeira a entrar no ambiente. E percebe isto, pois é a única que figura na lista de usuários. Ela lembra da instrução do professor e reconhece que o indicador [x] na frente de seu nome representa que ela tem o controle da ferramenta. Leandra não acha o sinal [x] intuitivo, pois se o professor não tivesse explicado, ela não saberia seu significado. Leandra também acha que este sinal deveria ser gráfico **[RQ18]**. Ela também percebe que todos os botões da barra de ferramentas estão habilitados, com exceção de um, que serve para solicitar o controle da ferramenta. Leandra então fica aguardando que seu colega entre no ambiente.

Matheus entra logo em seguida. Ele percebe que está sozinho no ambiente e por alguns minutos espera que Leandra entre no ambiente. Como nada acontece ele chama o professor. O professor então percebe, ao olhar para a barra de título da janela, que Matheus entrou no grupo errado e pede que ele saia da ferramenta e entre novamente. No entanto, agora ele deve entrar no grupo “cin” **[RQ19]**. Matheus questiona sobre a possibilidade de poder ver os grupos criados e poder escolher em qual grupo entrar ao invés de ter que lembrar o nome do grupo **[RQ14]**. Matheus segue as instruções do professor e entra corretamente no grupo indicado.

Ao entrar no grupo correto, Matheus vê através da lista de usuários que Leandra já se encontra no ambiente. Ele já esperava que Leandra tivesse o controle da ferramenta, pois lembra que o professor explicou que o primeiro a entrar no ambiente recebe automaticamente o controle. Ele também não se espanta ao ver quase todos os botões da ferramenta desabilitados, porque sabe que somente quem tem o controle da ferramenta é que pode manipular a área de trabalho compartilhada. Mas acha que poderiam ter outras opções que ele pudesse usar para ajudar na tarefa **[RQ20]**, ele tenta usar alguns botões, mas como estão desabilitados nada acontece **[RQ21]**.

Leandra resolve convidar Matheus para ler o primeiro problema, mas lembra da regra (c) e convida o colega através do *chat*. Ela acha que isso não seria necessário se o problema estivesse na tela da ferramenta **[RQ22]** e não no quadro como o professor colocou. Matheus aceita o convite e após alguns minutos, que cada um reservou para ler atentamente o problema, Matheus indaga Leandra sobre o que ela achou da questão. Leandra tem dificuldades de separar suas mensagens das mensagens de seu colega, pois todas têm a mesma formatação, sendo diferenciada somente pelo nome do autor no início da frase. Acostumada com o uso de ferramentas de *chat*, Leandra gostaria de poder personalizar a formatação de suas mensagens para facilitar a leitura **[RQ23]**. Leandra acha que melhor do que se expressar através do *chat* seria desenhar o diagrama. Matheus concorda prontamente. Então Leandra começa a desenhar.

Matheus assiste sua colega desenhar o diagrama, que representa a situação-problema, no ambiente compartilhado. Ele não concorda com o diagrama que ela está construindo e tenta se comunicar com ela. Mas como Leandra está concentrada montando o diagrama, ela não percebe o que ele escreve no *chat* **[RQ23]**.

Matheus tenta insistentemente chamar a atenção dela pela ferramenta, mas não encontra nenhuma outra opção a não ser o *chat* **[RQ20]**. Como última alternativa, Matheus resolve pedir o controle da ferramenta. Ele clica no botão para solicitar o controle e percebe na lista de usuário que aparece o indicador [r]

na frente do seu nome. Matheus não acha o sinal [r] intuitivo, pois se o professor não tivesse explicado, ele não saberia seu significado. Ele acha que este sinal poderia ser gráfico **[RQ18]**. Infelizmente Leandra também não percebe que o colega pediu o controle **[RQ24]**, pois está concentrada na resolução da tarefa.

Não tendo alternativa, Matheus levanta-se e fala em voz alta:

— Leandra, eu estou falando com você no *chat*.

Nesse momento ele percebe que quebrou uma das regras estipuladas pelo professor e imediatamente busca-o com seu olhar. E informa ao professor que não teve alternativa, pois Leandra não respondia ao *chat*. Leandra informa que não estava vendo as mensagens do colega e que deveria ter alguma forma de chamar sua atenção pela ferramenta **[RQ25]**.

Ao ver as mensagens do colega, Leandra relê o problema e reflete sobre o que estava fazendo. Ela percebe que não estava desenhando o diagrama correto. Pede desculpas ao colega por não ter visto isto antes. Informa que vai corrigir seu erro, mas que não vai lhe passar o controle, pois sabe qual o diagrama correto. Matheus concorda com Leandra e não tem mais interesse pelo controle, mas não sabe como cancelar sua requisição anterior, pergunta ao professor e ele informa que não há como fazer isso na ferramenta **[RQ26]**.

Leandra limpa a área compartilhada e começa a desenhar o novo diagrama. Enquanto ela desenha, Matheus concorda com o diagrama e vai acompanhando o desenrolar da tarefa concordando com cada passo realizado por sua colega.

Leandra termina de desenhar o diagrama e Matheus ao ver que a colega terminou de desenhar faz comentários através do *chat* para que ela perceba que ele está totalmente de acordo. Ele também infere a solução do problema e apresenta para ela através do *chat*. Leandra pega o resultado oferecido pelo colega e completa o diagrama, para demonstrar que concorda com a resposta dele. Matheus percebe que Leandra concorda quando a vê editar o diagrama para colocar a resposta sugerida por ele.

Para completar a tarefa, Leandra deixa explícito no *chat* que concorda com o colega e o convida para passar para o próximo problema. Ela acha que deveria ter alguma forma de encerrar o problema na ferramenta **[RQ27]**. Matheus concorda com sua colega e expressa isso claramente através do *chat*, mas ressalta que pretende ter o controle da ferramenta para resolver o próximo problema.

Leandra recebe a mensagem do seu colega e só neste momento percebe na lista de usuário o indicador [r] na frente do nome de Matheus. Isso indica que o colega solicitou o controle da ferramenta. Matheus aguarda que Leandra lhe passe o controle. Leandra então concorda em passar o controle da ferramenta para ele e pressiona o botão que realiza esta ação.

Após passar o controle, Leandra percebe que os botões da sua barra de ferramenta ficam quase todos desabilitados. Matheus recebe o controle e observa que os botões da sua barra de ferramentas ficaram habilitados e percebe na lista de usuários o indicador [x] na frente de seu nome, que representa que ele tem o controle da ferramenta. Ele limpa a área compartilhada para remover o diagrama do problema anterior e convida sua colega para ler o próximo problema.

7.1.7. REQUISITOS LEVANTADOS

A partir dos cenários descritos na seção anterior, foi possível derivar os seguintes requisitos adicionais que deveriam estar presentes no sistema (também indicados no roteiro negativo através de indicadores, como por exemplo, **[RQ14]**), os requisitos serão apresentados em dois níveis, conforme influenciam as funcionalidades da ferramenta Gérard ou da plataforma Plattus:

Tabela 7.1 – Requisitos levantados na análise de cenários (primeira iteração)

Requisito	Gérard	Plattus
[RQ14]	A ferramenta deve fornecer ao usuário uma lista que apresente em quais grupos o usuário pode conectar-se.	A plataforma de *groupware* deve fornecer para as ferramentas a lista de grupos disponíveis para conexão, juntamente com a lista

[RQ15]	A ferramenta deve permitir que o usuário visualize a lista de pessoas que estão conectadas ao grupo que ele escolheu.	de usuários conectados ao grupo. Adicionalmente, a plataforma pode prover um elemento de interface (*widgets*) que forneça um diálogo de entrada para os usuários nos ambientes, que apresente à quais grupos o usuário pode conectar-se e a lista de usuários conectados a cada grupo.
[RQ16]		A plataforma de *groupware* não deve permitir que dois usuários conectem com o mesmo login.
[RQ17]		A plataforma de *groupware* deve prover um mecanismo de segurança para autenticar a conexão do usuário.
[RQ18]	A interface da ferramenta deve prover uma forma mais intuitiva de exibir quem está com o controle da ferramenta.	A plataforma de *groupware* deve apresentar no elemento de interface (*widgets*) a lista de usuários de forma intuitiva, de preferência com elementos gráficos, chamando a atenção do usuário para eventos como a entrada e saída de usuários e o pedido de controle da ferramenta.
[RQ19]	A Ferramenta deve permitir que o usuário troque de grupo.	A plataforma de *groupware* deve permitir ao usuário trocar de grupo.

[RQ20]	Um usuário que não tem o controle da aplicação deve ter formas de participar mais ativamente do desenrolar da tarefa, por exemplo, fornecendo *feedbacks* aos colegas.	
[RQ21]	Quando o usuário tentar utilizar botões desabilitados, a ferramenta deve informá-lo que as funcionalidades não estão acessíveis porque ele não está com o controle da ferramenta.	
[RQ22]	A interface da ferramenta deve possuir uma área para exibir o enunciado dos problemas.	A plataforma de *groupware* deve permitir a definição antecipada de grupos associados a uma dada ferramenta, com os respectivos dados da sessão, problemas e usuários. E enviar estes dados às ferramentas no momento que o usuário conectar-se ao grupo.
[RQ23]	A interface do *chat* deve ser mais amigável e usável, dando mais opções de formatação e sons aos usuários, permitindo que eles interajam melhor, percebendo facilmente quando o outro está tentando se comunicar, obtendo com facilidade a atenção do outro e sabendo quem disse o que na conversa que está na tela.	A plataforma de *groupware* deve apresentar no elemento de interface (*widgets*) de *chat* elementos comuns às ferramentas de *chats* e mensagens instantâneas atuais.

[RQ24]	A ferramenta deve fazer o usuário perceber rapidamente quando o seu(s) parceiro(s) pede(m) o controle da ferramenta	A plataforma de *groupware* deve gerar eventos específicos para alertar as ferramentas de que um usuário pediu o controle da ferramenta. A plataforma de *groupware* deve apresentar no elemento de interface (*widgets*) a lista de usuários de forma intuitiva, de preferência com elementos gráficos, chamando a atenção do usuário para eventos como a entrada e saída de usuários e o pedido de controle da ferramenta.
[RQ25]	A ferramenta deve fornecer uma alternativa para que um usuário possa chamar a atenção de outro.	A plataforma de *groupware* deve gerar eventos específicos para alertar as ferramentas de que um usuário está tentando chamar atenção de outro.
[RQ26]	A interface da ferramenta deve fornecer um botão para cancelar uma solicitação prévia de controle	A plataforma de *groupware* deve permitir o cancelamento de solicitações de controle pelos usuários
[RQ27]	Deve haver na ferramenta uma forma explicita de informar a conclusão da resolução de um problema, passando para o problema seguinte.	A plataforma de *groupware* deve fornecer forma explicita de informar a conclusão da resolução de um problema, passando para o problema seguinte.

7.2. SEGUNDA ITERAÇÃO

Nesta segunda iteração, buscamos refinar ainda mais os requisitos levantados na primeira iteração. Para isso, montamos um cenário similar ao cenário da primeira iteração, mas utilizando um sistema hipotético que implementa todos os requisitos levantados anteriormente. Com isso o cenário estará funcionando como um protótipo de baixo custo que Carroll [CAR2000] chama de protótipos "leves" (*"soft" prototype*), o qual favorece a descoberta de novos requisitos e o refinamento dos requisitos já identificados.

Neste cenário, primeiramente apresentaremos uma nova e hipotética versão do sistema Gérard, que não foi implementada. Neste caso o cenário é o protótipo que atende aos requisitos levantados na iteração anterior. Para atender aos requisitos em nível do componente, significa que os requisitos associados na plataforma de *groupware* também foram atendidos.

Como esse cenário dedica uma seção para apresentar as características do novo sistema hipotético, não vimos necessidade de apresentar um roteiro normal de uso do sistema. Neste caso, apresentaremos apenas os roteiros positivos e negativos.

A seguir, apresentaremos textualmente as novas características do Gérard. No decorrer do texto quando uma referência do tipo **[RQ01]** aparecer, significa que um requisito levantado na iteração anterior foi atendido pelo protótipo.

7.2.1. PROTÓTIPO

Juntamente com a nova versão do Gérard, foi criada uma ferramenta para edição de problemas. Na ferramenta um professor pode criar novos problemas, ou editar problemas existentes, definindo o enunciado e desenhando o diagrama resposta para o problema. Os problemas criados são armazenados no formato XML, no próprio servidor da plataforma Plattus, no repositório de problemas. Os problemas do repositório podem ser utilizados por qualquer professor para definir atividades para serem resolvidas por seus alunos através do Gérard.

A plataforma de *groupware* Plattus foi integrada a um Sistema de Gerenciamento de Cursos (LMS - *Learning Management System*), para autenticação dos usuários e para definição de atividades curriculares: o módulo de autenticação de usuários acessa ao banco de dados de usuários do LMS. Somente usuários cadastrados no LMS podem utilizar o componente Gérard e outros componentes da plataforma **[RQ16] [RQ17]**.

Através da interface do LMS, o professor pode agendar atividades para serem realizadas por seus alunos **[RQ22]**. Para isso, o professor escolhe o componente, neste caso o Gérard, mas futuramente outros componentes estarão disponíveis para os professores. O professor define uma atividade informando a data e o horário agendado para atividade, o tempo de duração, seleciona os alunos que participarão da tarefa, seleciona os problemas a partir do repositório de problemas e pode definir se o *floor control* será *Request-and-Wait* como na versão anterior do Gérard ou se será predefinido.

No *floor control* predefinido, o professor pode definir qual participante terá o controle da ferramenta no início e associar os participantes a cada um dos problemas escolhidos. Assim quando o problema estiver sendo resolvido pelos alunos, terá o controle da aplicação o aluno que o professor definiu na preparação da atividade.

Ao executar o Gérard, será apresentada ao aluno uma caixa de diálogo chamada “Rol de entrada”, onde o aluno vê a lista de todas as salas[1] ativas **[RQ14]**. Ao selecionar uma sala o aluno visualiza quem já está naquela sala **[RQ15]**. O aluno pode entrar em uma sala já existente ou criar uma nova sala. Ao criar uma nova sala o aluno entra automaticamente nela. As salas criadas por alunos são chamadas salas temporárias, enquanto que as salas definidas pelo professor na preparação de atividades são chamadas salas persistentes.

Nas salas criadas pelos alunos, os problemas são buscados do repositório aleatoriamente e o *floor control* será *Request-and-Wait* como na versão anterior do Gérard. A cada problema resolvido o próximo problema é

[1] O conceito de Salas substitui o conceito de grupos da versão anterior do Gérard. (ver página 55)

aleatoriamente selecionado até que todos saiam da sala, quando a sala é removida do servidor.

Somente podem entrar em uma sala persistente os alunos definidos pelo professor para participar da atividade. Quando um aluno está no “Rol de entrada” ele pode ver as atividades pré-definidas para ele e se a atividade estiver em andamento ele será alertado para entrar na sala referente à atividade.

Com a possibilidade de criação de salas temporárias pelos alunos, pode ser que em um dado momento existam muitas salas ativas, neste caso o “Rol de entrada” dispõe de uma ferramenta de pesquisa que permite ao aluno procurar por um colega e descobrir em que sala ele se encontra.

Quando o aluno estiver em uma sala e quiser trocar de sala, ele pode facilmente retornar ao “Rol de entrada” e mudar de sala **[RQ19]**. Se o aluno estiver em uma sala temporária no momento agendado pelo professor para uma atividade ele será alertado sob o fato de existir uma atividade aguardando por ele e com um simples clique ele poderá sair da sala atual e entrar automaticamente na sala reservada pelo professor para atividade **[RQ19]**.

A nova interface do Gérard é constituída de quatro áreas principais: uma lista de usuários ativos, fornecida pela plataforma Plattus; um *chat* para a comunicação dos usuários, também fornecido pela plataforma; uma área de trabalho compartilhada, utilizada para o desenho dos diagramas e uma área para apresentação do enunciado do problema ativo.

A lista de usuários apresenta ícones gráficos para fornecer aos alunos na sala o status dos usuários. Um ícone de um cadeado é apresentado junto ao nome do usuário que tem o controle da ferramenta **[RQ18]** e o ícone de uma chave é apresentado ao lado do nome dos usuários que requisitaram o controle da ferramenta. Dentro do ícone da chave um número indica a posição do usuário na fila de solicitação. Sempre que um aluno solicita o controle da ferramenta um som é emitido e aparece imediatamente a chave ao lado de seu nome, que fica girando por alguns segundos, então ela estabiliza e passa a apresentar a posição do aluno na fila **[RQ24]**.

Para as salas persistentes, os nomes dos alunos definidos pelo professor para a atividade e que ainda não entraram na sala sâo apresentados na cor cinza.

Sempre que um aluno entra ou sai da sala, é emitido um som que indica a entrada ou saída de alunos. Quando o aluno entra, além do som seu nome fica em destaque por alguns segundos permitindo aos outros perceberem quem entrou. E quando um usuário sai seu nome vai desaparecendo levemente para também para ajudar na percepção.

Ao lado do nome de cada aluno na lista de usuários, existe também um ícone de um sino, que pode ser utilizado para chamar a atenção do aluno. Quando se clica no sino, a tela do aluno vibra juntamente com a emissão de um som de sinos tocando, que buscam chamar sua atenção. Na lista de usuários o sino do aluno que lhe chamou a atenção pisca por alguns segundos para que o aluno perceba quem está lhe chamando a atenção **[RQ25]**.

O *chat* para conversação entre os alunos agora dispõe de recursos de formatação (fontes, tamanho de letra, negrito, itálico, cor, etc...). Agora, sempre no envio de uma nova mensagem um som é emitido para que o usuário perceba a chegada da mensagem **[RQ23]**.

A tela principal do Gérard tem agora uma nova região dedicada à apresentação do problema que está sendo resolvido **[RQ22]**. Ao lado da descrição do problema existe um botão para finalização do problema **[RQ27]**. Somente o aluno que tem o controle pode indicar a conclusão do problema, clicando no botão, mas após isso todos os outros alunos serão interrogados se concordam com a conclusão do problema e sendo a resposta positiva o problema é concluído e o próximo problema será apresentado.

Na barra de ferramentas do Gérard existem novos botões. Um novo botão foi adicionado para permitir aos alunos cancelarem um pedido de controle **[RQ26]**. Quando pressionado o usuário abdica de uma solicitação de controle, um som é emitido e a chave que se encontra em frente ao seu nome gira por alguns segundos até desaparecer levemente.

Um novo botão com um ícone de balão de fala permite que um aluno possa fazer comentários (*feedback*) diretamente na área de trabalho compartilhada **[RQ20]**. Qualquer usuário pode selecionar um objeto na área compartilhada e apresentar comentários. O comentário será visualizado na própria área compartilhada dentro de um balão explicativo e lá permanecerá por alguns segundos, podendo ser fechado pelo usuário antes do final do tempo.

Ao invés de apresentar mensagens aos usuários que tentem usar botões desabilitados, nesta nova versão o Gérard apresenta apenas os botões que o usuário pode usar ocultando os botões que não podem ser utilizados **[RQ21]**.

7.2.2. ATORES

Os Alunos Leandra e Matheus e o professor.

7.2.3. CONTEXTO

O professor pretende testar o novo sistema e convida os alunos Leandra e Matheus para uma nova sessão de utilização do sistema. O professor agenda a atividade para a próxima quarta-feira às 16h, mas Leandra informa que não pode ir à universidade neste dia, pois estará na casa de seus pais que moram em outra cidade. Como na casa da Leandra tem um computador com acesso à Internet, eles acordam que Leandra acessará o sistema a partir dele. O professor entrega a Leandra um CD de instalação para que ela instale o Gérard no computador.

7.2.4. AMBIENTE (SETTING)

O professor prepara com dois dias de antecedência a atividade a ser realizada pelos alunos, através do LMS. Para isso, ele escolhe o componente, neste caso o Gérard. Ele define a data, o horário de início da atividade, e o tempo para realização da atividade que foi de meia hora. Em seguida, o professor adiciona à atividade seu próprio usuário e os dois alunos, Leandra e Matheus, e seleciona dois problemas que deverão ser resolvidos pelos alunos. Finalmente, o

professor define o *floor control* como predefinido e associa o primeiro problema à Leandra e o segundo problema para Matheus.

Os problemas escolhidos pelo professor foram:

1) Mariana e Túlio encontraram conchinhas na praia. Mariana achou 21 e Túlio achou 16. Quantas conchinhas Mariana achou a mais que Túlio?

2) João tinha 63 bolas de gude. Jogando com seus colegas perdeu 31 bolas. Quantas João tem agora?

Matheus e o professor estão no laboratório, que tem 12 computadores dispostos em uma grande bancada que atravessa a sala em quase toda sua extensão. Os computadores na bancada estão voltados em direções opostas permitindo um melhor aproveitamento do espaço do ambiente. Todos os computadores estão ligados a uma rede local, que permite também acesso à Internet.

Leandra está em sua casa, usando seu computador pessoal conectado à Internet. O Gérard já foi devidamente instalado por ela, através do CD de instalação fornecido pelo professor.

7.2.5. ROTEIRO POSITIVO

Ao executar o Gérard, Matheus vê a tela do “Rol de entrada” e após autenticar-se com o mesmo *login* e senha que utiliza no LMS, vê a lista de todas as salas ativas. Como ainda é muito cedo, Matheus entra em uma das salas criada por outro aluno e aproveita para praticar um pouco.

Próximo ao horário da atividade, Matheus recebe uma mensagem informando que é esperado para uma atividade curricular, mostrando o tempo que falta para o início da atividade e perguntando se ele deseja entrar nesta sala. Matheus gosta da funcionalidade e responde que sim. Automaticamente ele entra na referida sala.

Ao entrar na sala ele percebe, na lista de usuários, que os nomes do seu professor e Leandra estão na cor cinza, informando que eles também são

esperados na sala, mas ainda não chegaram. Ele também percebe que mesmo sem Leandra está on-line, ao lado de seu nome existe um cadeado, também cinza, indicando que a mesma tem o controle da ferramenta. O problema ainda não aparece na área de problemas, porque ainda não é o horário marcado para o início da atividade. Matheus percebe também que na sua tela há poucos botões e que todos estão habilitados.

O professor acompanhou as ações de Matheus para entrar no ambiente e em seguida liga outro computador e também executa o Gérard. O professor também é convidado a entrar na sala e prontamente aceita. O professor e Matheus conversam um pouco através do *chat*.

Não demora muito e Leandra também entra na sala. Matheus e o professor percebem que ela entrou, pois quando Leandra entra, além do som, seu nome fica em destaque por alguns segundos permitindo aos outros perceberem que ela entrou. Também, tanto seu nome como o ícone do cadeado deixam a cor cinza e passam a ser apresentados normalmente.

O momento marcado para o início da atividade chega e o problema é apresentado para todos na região dedicada para isso. O professor convida todos para lerem o problema e diz que está ali para acompanhar o desenrolar da tarefa e que pouco intervirá na atividade. Informa também que as regras continuam as mesmas da interação anterior.

Leandra lê o problema e pergunta para Matheus pelo *chat* se ele já leu o problema. Matheus diz que sim e pergunta se ela concorda que o problema é de comparação. Ela concorda e informa que vai desenhar o diagrama. Matheus percebe que a colega não está desenhando o diagrama corretamente e resolve utilizar o novo recurso de *feedback*.

Matheus clica no botão de *feedback* e no diagrama que Leandra está desenhando. Imediatamente abre uma janela onde ele escreve seu comentário. Quando Matheus confirma seu comentário, em todos os computadores que participam da atividade, aparece um balão explicativo com a mensagem de

Matheus apontando para o diagrama. Como o olhar de Leandra está voltado para esta região, ela vê a mensagem, fecha o balão e corrige o diagrama.

Matheus, ao vê que a colega terminou de desenhar, faz comentários através do *chat*. Ele também infere a solução do problema e apresenta para ela através do *chat*. Leandra pega o resultado oferecido pelo colega e completa o diagrama, para demonstrar que concorda com a resposta dele. Matheus vê a colega editar o diagrama para adicionar a resposta sugerida por ele.

Para completar a tarefa, Leandra clica no botão ao lado da descrição do problema, que serve para finalizar a resolução do problema. Neste momento Matheus e o professor recebem um mensagem informando que Leandra considera o problema resolvido e pedindo a confirmação da conclusão do problema. Ambos concordam com o diagrama desenhado por Leandra e confirmam a conclusão da solução.

Neste momento, a descrição do novo problema é apresentada na tela, automaticamente a área de trabalho compartilhada é limpa e o controle da ferramenta passa para Matheus. Leandra recebe uma mensagem informando que o controle foi passado para Matheus obedecendo às regras definidas na preparação da atividade. Leandra percebe que Matheus está com o controle, pois vê o cadeado ao lado de seu nome.

7.2.6. ROTEIRO NEGATIVO

Os requisitos aparecem mais facilmente no roteiro negativo, pois é o que requer uma visão crítica sobre a interação dos usuários com o sistema. No decorrer do texto, quando uma referência do tipo **[RQ01]** aparecer, significa que um requisito foi levantado. Todos os requisitos levantados nesta seção serão apresentados em uma tabela na próxima seção.

Ao executar o Gérard, Matheus vê a tela do “Rol de entrada” e após autenticar-se com o mesmo *login* e senha que utiliza no LMS, vê a lista de todas as salas ativas. Como ainda é muito cedo, Matheus entra em uma das salas criada por outro aluno e aproveita para praticar um pouco.

Próximo ao horário da atividade Matheus recebe uma mensagem informando que é esperado para uma atividade curricular, mostrando o tempo que falta para o início da atividade e perguntando se ele deseja entrar nesta sala. Matheus gosta da funcionalidade e responde que sim. Automaticamente ele entra na referida sala.

Ao entrar na sala ele percebe, na lista de usuários, que os nomes do seu professor e Leandra estão na cor cinza, informando que eles também são esperados na sala, mas ainda não chegaram. Ele também percebe que mesmo sem Leandra está on-line, ao lado de seu nome existe um cadeado, também cinza, indicando que a mesma tem o controle da ferramenta. O problema ainda não aparece na área de problemas, porque ainda não é o horário marcado para o início da atividade. Matheus percebe também que na sua tela há poucos botões e que todos estão habilitados. Mas ele nada pode fazer, porque a atividade ainda não começou.

O professor acompanhou as ações de Matheus para entrar no ambiente e em seguida liga outro computador e também executa o Gérard. O professor também é convidado a entrar na sala e prontamente aceita. O professor e Matheus conversam um pouco através do *chat*.

Ao conversar no *chat* o professor verificou que o tipo de fonte e as cores que ele havia configurado em uma sessão anterior não haviam sido gravadas e não gostou de ver que tudo voltou à formatação padrão. Ele gostaria que as definições feitas por ele na ferramenta pudessem permanecer mesmo entre sessões diferentes e até entre computadores diferentes, pois cada vez que utiliza o sistema, o faz em computadores diversos **[RQ28]**. Então ele configura novamente a fonte e cores de sua preferência.

Até a hora marcada para o início da atividade Leandra não chega à sala. Então a plataforma automaticamente passa o controle do sistema para Matheus. Seguindo a lógica que o aluno que deveria ter o controle da ferramenta pode não vir e o andamento da tarefa poderia ser prejudicado se o sistema continuasse bloqueado aguardando este aluno.

Matheus está lendo o problema, quando Leandra chega. Matheus e o professor percebem que ela entrou, pois quando Leandra entra além do som seu nome fica em destaque por alguns segundos permitindo aos outros perceberem que ela entrou. Também, seu nome deixa a cor cinza e passa a ser apresentado normalmente. Mas ela não recebe o controle da ferramenta.

O professor gostaria de ter o direito de tirar o controle de Matheus e entregar para Leandra, mas não existe distinção entre os usuários e nenhum usuário tem esse direito **[RQ29]**. Então ele pede a Leandra que solicite o controle da ferramenta e a Matheus que passe o controle à colega.

Quando Leandra solicita o controle da ferramenta, um som é emitido e no computador de todos os participantes. Aparece imediatamente a chave ao lado de seu nome, que fica girando por alguns segundo, então ela estabiliza e passa a apresentar em seu interior o número 1 (um), que indica que a mesma é a primeira a receber o controle quando ele for liberado. Matheus passa o controle para Leandra.

Então o professor convida todos para lerem o problema e diz que está ali para acompanhar o desenrolar da tarefa e que pouco intervirá na atividade. O professor reflete que não poderá participar de todas as atividades que programar. Ele gostaria de ter a possibilidade de ver o desenrolar da atividade em outro momento **[RQ30]**.

O Professor Informa que as regras continuam as mesmas da interação anterior. Mateus pergunta ao professor porque as regras não estão no ambiente, assim poderiam ser consultadas quando necessário **[RQ31]**. O professor informa que uma delas já faz parte do ambiente, que é a concordância da conclusão do problema.

Leandra lê o problema e pergunta para Matheus pelo *chat* se ele já leu o problema. Matheus diz que sim e pergunta se ela concorda que o problema é de comparação. Ela concorda e informa que vai desenha o diagrama. Matheus percebe que a colega não está desenhando o diagrama corretamente e resolve utilizar o novo recurso de *feedback*.

Matheus clica no botão de *feedback*, clica no diagrama que Leandra está desenhando. Imediatamente abre uma janela onde ele escreve seu comentário. Quando Matheus confirma seu comentário, em todos os computadores que participam da atividade, aparece um balão explicativo com a mensagem de Matheus apontando para o diagrama.

Infelizmente, neste momento a mãe de Leandra a chama e leandra deixa o quarto, não vendo o feedback do colega **[RQ32]**. Matheus acha estranho, pois não vê nenhuma reação de sua colega ao seu comentário. Ele acha que ela está refletindo sobre o que ele escreveu. O tempo definido para o balão escoa e ele desaparece lentamente. E Matheus acha estranho que sua colega não atende aos seus apelos.

Matheus pergunta ao professor se existe a possibilidade de enviar *feedback* sonoro, pois seu comentário foi bastante longo e talvez fosse melhor entendido se fosse falado ao invés de escrito **[RQ33]**. Pergunta também sobre a possibilidade do uso de voz e vídeo na comunicação **[RQ34]**, além do *chat*. O professor informa que não existe essa possibilidade, mas que são idéias interessantes.

O professor gostaria de enviar uma mensagem de *chat* para Leandra fazendo algumas colocações, mas não gostaria que Matheus visse **[RQ35]**. Ele sabe que qualquer mensagem enviada no *chat* é vista por todos. Então ele resolve aguardar mais um pouco, se nada acontecer ele vai terá que fazer o comentário publicamente.

Leandra chega, mas como o feedback já desapareceu, ela não sabe nada sobre os comentários de seu colega **[RQ36]**. Matheus resolve chamar sua atenção e clica no ícone do sino ao lado de seu nome na lista de usuários. A tela de Leandra vibra juntamente com a emissão do som de sinos tocando, que buscam chamar sua atenção. Na lista de usuários o sino ao lado no nome de Matheus pisca por alguns e Leandra vê que Matheus quer falar com ela.

Ela pergunta, pelo *chat*, o que ele quer. Ele pergunta por que ela ignorou seus comentários. Leandra informa que não estava na frente do

computador e não viu nenhum comentário. Matheus refaz seu comentário. Quando Matheus confirma seu comentário, em todos os computadores que participam da atividade, aparece um balão explicativo com a mensagem de Matheus apontando para o diagrama. Como o olhar de Leandra está voltado para esta região, ela vê a mensagem, fecha o balão e corrige o diagrama.

Leandra termina de desenhar o diagrama e Matheus ao vê que a colega terminou de desenhar faz comentários através do *chat*. Ele também infere a solução do problema e apresenta para ela através do *chat*. Leandra pega o resultado oferecido pelo colega e completa o diagrama, para demonstrar que concorda com a resposta dele. Matheus vê a colega editar o diagrama para adicionar a resposta sugerida por ele.

Para completar a tarefa Leandra clica no botão ao lado da descrição do problema, que serve para finalizar a resolução do problema. Neste momento Matheus e o professor recebem uma mensagem informando que Leandra considera o problema resolvido e pedindo a confirmação da conclusão do problema. Ambos concordam com o diagrama desenhado por Leandra e confirmam a conclusão da solução.

Neste momento, a descrição do novo problema é apresentada na tela, automaticamente a área de trabalho compartilhada é limpa e o controle da ferramenta passa para Matheus. Leandra recebe uma mensagem informando que o controle foi passado para Matheus obedecendo às regras definidas na preparação da atividade. Leandra percebe que Matheus está com o controle, pois vê o cadeado ao lado de seu nome.

7.2.7. REQUISITOS LEVANTADOS

A partir dos cenários descritos na seção anterior, foi possível derivar os seguintes requisitos adicionais que deveriam estar presentes no sistema (também indicados no roteiro negativo através de indicadores, como por exemplo, **[RQ28]**). Quando um novo indicador aparecer na descrição do requisito (por exemplo, o indicador **[RQ37]** apresentado no texto explicativo do requisito **[RQ28]),** significa

que um novo requisito surgiu derivado do requisito que está sendo apresentado e que o mesmo será apresentado mais abaixo na tabela.

Como na seção anterior, os requisitos serão apresentados em dois níveis, conforme influenciam as funcionalidades da ferramenta Gérard ou da plataforma Plattus:

Tabela 7.2 – Requisitos levantados com a análise de cenários (segunda iteração)

Requisito	**Gérard**	**Plattus**
[RQ28]	A ferramenta deve armazenar as preferências de configurações dos usuários, não somente do *chat*, mas quaisquer serviços configuráveis oferecidas ao usuário. Estas preferências devem persistir mesmo entre sessões diferentes e até entre computadores diferentes.	A plataforma de *groupware* deve prover um repositório persistente centralizado de perfis de usuários. O perfil será composto de páginas. Cada página armazena os dados de preferências do usuário em uma determinada ferramenta. Cada ferramenta só poderá ler e gravar informações da página reservada para ela **[RQ37]**.
[RQ29]	Diferentes tipos de usuários devem ter privilégios diferenciados nas ações que podem realizar na ferramenta. Principalmente o professor ou um tutor que podem ser responsáveis pela coordenação das atividades.	A plataforma de *groupware* deve permitir a definição de papéis para os usuários que participam de uma atividade compartilhada.[1] Ao definir uma atividade o professor pode definir os papéis que serão assumidos pelos usuários na execução da mesma. Exemplos de papéis são: Aluno,

[1] Este requisito foi levantado na análise de competidores, mas não foi implementado no protótipo. Veja o requisito [RQ10] levantado na análise de competidores na página 51.

		professor, moderador, coordenador, etc. O papel assumido pelo usuário será enviado à ferramenta no momento de conexão do usuário, cabendo a ferramenta decidir quais as ações e privilégios estão associados a cada papel.
[RQ30]		Todos os passos de atividade realizada em uma sala persistente devem ser armazenados no servidor para posterior visualização **[RQ38]**.
[RQ31]	Todas as informações importantes para execução das tarefas devem está disponíveis ou incorporadas na interface da ferramenta. Este é o caso das regras que devem ser seguidas para realização da atividade. As informações podem ser providas em diversos formatos como texto, vídeos, animações, etc.	Na definição de uma atividade a plataforma de *groupware* deve permitir o armazenamento, além do XML que define a atividade, de diversos tipos de arquivos associados à execução da atividade. Esses arquivos serão enviados para ferramenta juntamente com o XML da atividade. Cabe a ferramenta conhecer os formatos dos arquivos e a forma de apresentação dos mesmos aos usuários.

[RQ32]	A ferramenta deve prover aos usuários uma forma de definir status de disponibilidade, como por exemplo, *online* ou ausente. Que permita aos outros usuários identificar sua disponibilidade durante a realização da tarefa.	A plataforma de *groupware* deve prover vários status de disponibilidade e privacidade. Permitindo aos usuários informar aos outros sua disposição ou possibilidade para colaborar em um dado momento.
[RQ33] **[RQ34]**	A ferramenta pode prover aos usuários a possibilidade de fornecer *feedbacks* sonoros. Neste caso, ao invés de balões explicativos, um ícone de um auto-falante ao lado do objeto no ambiente compartilhado pode representar o feedback. Neste caso bastará ao usuário clicar no ícone para ouvir o comentário. A ferramenta deve prover aos usuários a possibilidade do uso de voz e vídeo na comunicação.	A plataforma de *groupware* deve prover também comunicação não textual como som e vídeo.
[RQ35]	A ferramenta deve prover aos usuários a possibilidade de comunicarem-se de forma privativa.	A plataforma de *groupware* deve prover aos usuários a possibilidade de comunicarem-se de forma privativa.
[RQ36]	A ferramenta deve manter as mensagens de *feedback* na tela até que o usuário indique que recebeu (leu ou ouviu).	

[RQ37]	Para garantir a implementação do requisito [RQ28], toda ferramenta ao conectar-se com a plataforma de *groupware* deve fornecer seu IC (Identificador de Componente) e a versão da ferramenta.	Só poderão conectar-se à plataforma de *groupware* os componentes que tenham um identificador IC (Identificador de Componente) válido no servidor de *groupware*. A partir do IC a plataforma pode selecionar a página correta pertencente à ferramenta no perfil do usuário e fornecer o acesso aos dados nela contidos.
[RQ38]	Em salas persistentes o desenvolvedor da ferramenta poderá fornecer um visualizador que permita assistir o desenrolar da atividade realizada na sala pelo professor em outro momento.	

7.3. CONSIDERAÇÕES FINAIS

A técnica de cenários é muito flexível, podendo ser adaptada pelos projetistas de acordo com seus interesses específicos. Nossa metodologia baseou-se no desenvolvimento de cenários caricaturados de uma situação de uso da ferramenta Gérard.

O uso de cenários de forma iterativa, ou seja, quando o cenário seguinte incorpora ao sistema as funcionalidades descritas pelos requisitos do cenário anterior, permite a utilização de cenários como protótipos "leve" ("*soft*" *prototype* – [CAR2000]), trazendo baixo custo e maior rapidez no processo de desenvolvimento do sistema.

Foi interessante notar que um requisito levantado na análise de competidores e que não foi implementado no protótipo, reapareceu na análise de cenários. O requisito [RQ10] indicava a necessidade de definição de diferentes papéis para diferenciar as possibilidades ações dos usuários na execução de tarefas. Esse requisito não foi implementado no protótipo por uma questão de simplificação da prototipação, porém, a análise dos cenários reforçou a necessidade deste requisito em sistemas colaborativos educacionais, quando o reapresenta no requisito [RQ29], levantado na análise do segundo cenário.

Neste trabalho, realizamos apenas duas iterações com cenários, mas isso pode ser repetido várias vezes, pois cada nova iteração refina-se os requisitos levantados no cenário anterior e levantam-se novos requisitos. Além disso, cenários facilitam a comunicação entre desenvolvedores e usuários, pois fornecem uma linguagem mais natural, baseada em uma história de utilização do futuro sistema.

8. CONCLUSÕES E TRABALHOS FUTUROS

Componentes síncronos de aprendizagem são sistemas que visam apoiar o aprendizado colaborativo fornecendo um conjunto de objetos compartilhados, estes formam o espaço compartilhado que representa o ambiente virtual para interação entre os alunos.

Desenvolver um sistema deste tipo foi um desafio que buscou nas necessidades e expectativas dos usuários as respostas para qual arquitetura e quais serviços deveriam ser oferecidos para resolver os principais problemas dos sistemas colaborativos: comunicação, consciência de colaboração, manutenção do espaço compartilhado, controle de acesso ao espaço compartilhado, definição de papeis e percepção (*awareness*). Porém entender e traduzir as necessidades dos usuários para arquétipos de *software* não foi uma tarefa fácil.

Uma característica importante do uso da análise de competidores foi que durante esta fase, nossa preocupação estava mais voltada para as estruturas arquiteturais e soluções de software utilizadas em cada produto. De uma forma quase que inconsciente os usuários pareciam algo distante que deveriam adaptar-se as soluções já bem estabelecidas no mercado.

As primeiras preocupações com as futuras reações dos usuários só começaram a aparecer na fase de desenvolvimento dos protótipos, mas foi durante a análise dos cenários que estas preocupações se tornaram mais evidentes. Pois quem constrói um cenário assume o papel do usuário, passa a defendê-lo e não mais ao sistema – uma característica natural dos desenvolvedores – tornando-se mais crítico em relação ao funcionamento do sistema, principalmente nos cenários negativos.

Percebemos que o cenário negativo permitiu-nos derivar, de forma muito natural, todos os requisitos identificados. Acreditamos, portanto, que o cenário positivo torna-se dispensável para tal propósito, sendo mais interessante na identificação e destaque dos pontos fortes do sistema, o que faz parte de nossos planos futuros.

O protótipo da plataforma de *groupware* Plattus foi construído com um *toolkit*, oferecendo seus serviços ao programador através de uma API, que não é a melhor alternativa quando se está trabalhando com uma linguagem orientada a objetos como Java. Como trabalho futuro pretende-se evoluir a plataforma para um *framework*, mais adequada ao estilo de programação que adotamos.

Atualmente encontra-se em desenvolvimento um novo componente baseado na plataforma Plattus. Este objetiva a disponibilização de um ambiente colaborativo síncrono, que possibilite a realização de experimentos de Física colaborativamente. Este trabalho está sendo desenvolvido por uma colega de mestrado, Maria Cláudia Barbosa Costa, e faz parte de um projeto de extensão coordenado pela CECINE – Coordenadoria do Ensino de Ciências do Nordeste, coordenado pelo Professor Hélio Teixeira Coelho. Esse projeto visa realizar cursos de extensão, cada um com 40 h de duração, nas áreas de Introdução à Astronomia, Introdução à Física, Introdução à Química, Introdução à Biologia, Introdução à Filosofia, e Introdução à Matemática, em regime semi-presencial de ensino, em cinco cidades do estado de Pernambuco. Estas cidades representam pólos principais de influência sócio-econômicas e culturais do estado de Pernambuco. Para cada curso espera-se um publico total de cem (100) pessoas por cidade. Para todas as cidades e todos os cursos, deveremos ter 3000 participantes no total. Esses cursos visam promover a formação inicial e continuada de estudantes do ensino médio, superior e pós-graduação orientando-os na compreensão de conceitos fundamentais da área.

REFERÊNCIAS BIBLIOGRÁFICAS

[ABD1997] Abdel-Wahab, H.; Kvande, B.; Kim, O.; Favreau, J. P. (1997). An Internet Collaborative Environment for Sharing Java Applications. Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS '97), p. 112-117.

[ALV2004] Alves, E. C.; Gomes, A.; Braga, M. (2004). Componentes de Aprendizagem Síncronos. XV Simpósio Brasileiro de Informática na Educação – SBIE, p. 533-540.

[ALV2005] Alves, S. V. L. (2005). Percepção em sistemas colaborativos síncronos de aprendizagem. Universidade Federal de Pernambuco. Relatório técnico.

[BØD2000] Bødker, S. (2000). Scenarios - setting the stage for reflection and action in user-centered design. Interacting with computers, v. 13, n. 1, p. 61-77.

[BOR2000] Borchers, J. (2000). A Pattern Approach to Interaction Design. Proceedings of the International Conference on Designing Interactive Systems, ACM Press, p. 369–378.

[BOY1993] Boyd, J. (1993). Floor control policies in multi-user applications. INTERACT '93 and CHI '93 conference companion on Human Factors in Computing Systems, p. 107-108.

[BRI1995] Bricker, L.; Tanimoto S.; Rothenberg, A.; Hutama, D.; Wong, T. (1995). Multiplayer Activities Which Develop Mathematical Coordination. Proceedings CSCL'95, ACM Press, p. 32-39.

[BRO1987] Brooks, F. P. (1987). No silver bullet: Essence and accidents of Software Engineering. IEEE Computer, v. 20, n. 4, p. 10-19.

[CAR1998] Carroll, J. M.; Rosson, M.; Chin, G.; Koenemann, J. (1998). Requirements

Development in Scenario-Based Design. IEEE Transactions on Software Engineering, v. 24, n. 12, p. 1156-1170.

[CAR2000] Carroll, J. M. (2000). Five Reasons for Scenario-Based Design. Interacting with Computers, v. 13, n. 1, p. 43-60.

[DAY1997] Day, M. (1997). What Synchronous Groupware Needs: Notification Services. Proceedings of the 6th Workshop on Hot Topics in Operating Systems, p. 118-122.

[DIL1996] Dillenbourg, P.; Baker, M; Blaye, A; O'Malley, C. (1996). The evolution of research on collaborative learning. In: E. Spada & P. Reiman (Eds) Learning in Humans and Machine: Towards an interdisciplinary learning science, p. 189-211.

[DOU1992] Dourish, P.; Belloti, V. (1992). Awareness and Coordination in Shared Workspaces. In: Proceedings of The Conference on Computer Supported Cooperative Work - CSCW'92October 1992, ACM Press, p. 107-114.

[ELL1991] Ellis, C.; Gibbs, S.; Rein, G. (1991). Groupware: Some issues and experiences. Communications of the ACM, v. 34, n. 1, p. 38-58.

[FER2003] Ferre, X. (2003). Integration of Usability Techniques into the Software Development Process. Proceedings of the International Conference on Software Engineering, p. 28-35.

[GAM1994] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley: Massachusetts.

[GAR2001] Lopez, G. P.; Skarmeta, A. G.; Rallo, R. (2001). ANTS: A new Collaborative Learning Framework. Proceedings of the European Conference on Computer-Supported Collaborative Learning, p. 22-24.

[GAR2002] Lopez, P. G; Skarmeta, A. G.; Pinyol, O. M.; Gavaldà, C. P.; Rallo, R. (2002). MOVE: component groupware foundations for collaborative virtual environments. Proceedings of the 4th international conference on Collaborative virtual environments, ACM Press, ISBN 1-58113-489-4.

[GRE1994] Greenberg, S.; Marwood, D. (1994). Real time groupware as a distributed system: concurrency control and its effect on the interface. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, p. 207-217.

[GUE2001] Guerrero, L.; Fuller, D. (2001). A Pattern System for the Development of Collaborative Applications. Information and Software Technology, v. 43, n. 7, p. 457-467.

[GUT1995] Gutwin, C.; Stark, G.; Greenberg, S. (1995). Support for Workspace Awareness in Educational Groupware. Proceedings of Computer Supported Collaborative Learning, ACM Press, p. 147-156.

[HIL1988] Hiltz, S. R. (1988). Collaborative learning in a virtual classroom: Hightlights of findings. Proceedings of the CSCW, p. 282-290.

[HSI1994] Hsia, P.; Samuel, J.; Gao, J.; Kung, D.; Toyoshima, Y.; Chen, C. (1994). Formal Approach to Scenario Analysis. IEEE Software, v. 11, n. 2, p. 33-41.

[JOH1992] Johansen, R. (1992). An introduction to computer augmented teamwork. In: Computer augmented teamwork: A guided tour.

[KOS1992] Koschmann, T. (1992). Computer support of collaborative learning. ACM SIGCUE Outlook, v. 21, n. 3, p. 1-3.

[KOS1993] Koschmann, T.; Newman, D.; Woodruff, E.; Pea, R.; Rowley, P. (1993). Technology and pedagogy for collaborative problem solving as a

context for learning. Report on a CSCW'92 workshop, SIGCHI Bulletin, v. 25, n. 4, p. 57-60.

[KRA1988] Krasner, G. E.; Pope, S. T. (1988). A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk. Journal of Object Oriented Programming, v. 1, n. 3, p. 26-49.

[KUM1996] Kumar, V. S. (1996). Computer-Supported Collaborative Learning: Issues for Research. Disponível em: <http://www.cs.usask.ca/grads/vsk719/academic/890/project2/project2.html>. Acesso em: 15 Mar. 2005.

[LIP2002] Lipponen, L. (2002). Exploring foundations for computer-supported collaborative learning. Proceedings of the Computer-supported Collaborative Learning, p. 72-81

[LIS1998] Li, S. F.; Hopper, A. (1998). A Framework to Integrate Synchronous and Asynchronous Collaboration. Proceedings of the 7th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, p. 96-103.

[MAN1997] McManus, M. M. (1997). Computer Supported Collaborative Learning. In: SIGGROUP Bulletin, v. 18, n. 1, p. 7-9.

[MEN2002] Mendes, A. (2002). Arquitetura de software: Desenvolvimento orientado para arquitetura. Rio de Janeiro: Campos, 2002. ISBN 85-352-1013-X

[MEZ2002] Menezes, G. G. (2002). O paradigma CSCL e a Avaliação discente mediada pelas NTICS: Reflexões através do conceito de contradições da teoria da Atividade. Dissertação de Mestrado - Programa de Pós-Graduação em Tecnologia, CEFET-PR, Curitiba-PR.

[NCS2005] NCSA Habanero. Disponível em: <http://www.isrl.uiuc.edu/isaac/Habanero/>. Acesso em: 04 Jul. 2005.

[ORO2004] Orozco, P. M.; Asensio, J. I.; Garcia, P.; Pairot, C.; Dimitriadis, Y. A. (2004). A decoupled architecture for action-oriented coordination and awareness management. Proceedings of the X International Workshop on Groupware (CRIWG'04), p. 246-261.

[PAD2003] Pádua, W. (2003). Engenharia de Software: Fundamentos, Métodos e Padrões. 2. ed. Rio de Janeiro: LTC.

[PAL1994] Palmer, J.; Fields, N. (1994). Guest editor's introduction: Computer supported cooperative work. IEEE Computer, v. 27, n. 5, p. 15-17.

[PAT1990] Patterson, J.; Hill, R.; Roholl, S.; Meeks, S. (1990). Rendezvous: An Architecture for Synchronous Multi-User Applications. Proceedings of ACM Conference on Computer-Supported Cooperative Work, p. 317-328.

[PAT1991] Patterson, J. F. (1991). Comparing the Programming Demands of Single-user and Multi-user Applications. Proceedings of the 4th ACM SIGRAPH Conference on User Interface Software and Technology, p. 79-86.

[PAT1995] Patterson, J. F. (1995). A taxonomy of architectures for synchronous groupware applications. ACM SIGOIS Bulletin Special Issue: Papers of the CSCW'94 Workshops, v. 15, n. 3, p. 27-29.

[PAT1996] Patterson, J. F.; Day, M.; Kucan, J. (1996). Notification Servers for Synchronous Groupware. Conference on Computer Supported Cooperative Work, p. 122-129.

[PHI1999] Phillips, W. G. (1999). Architectures for Synchronous Groupware. Technical Report 1999-425, Queen's University, Kingston, Ontario, Canada. Disponível em: <http://symbiosis.rmc.ca/pub/arch-for-sync-groupware-tr1999-425.html>. Acesso em: 10 Mai. 2005.

[RIE1992] Riel, M. (1992). Cooperative learning through telecommunications. ACM SIGCUE Outlook, v. 21, n. 3, p. 14-17.

[ROB1998] Robinson, M.; Pekkola, S.; Snowdon, D. (1998). Cat's Cradle: Working with other people in overlapping real and virtual worlds through tangled strands of visual and other media. Proceedings of the 4th International Workshop on Groupware - CRIWG '98, p. 3-20.

[ROS1996] Roseman, M.; Greenberg, S. (1996). Building Real Time Groupware with GroupKit, A Groupware Toolkit. March. ACM Transactions on Computer Human Interaction, ACM Press, v. 3, n. 1, p. 66-106.

[ROT1998] Roth, J. (1998). CSCW Internet Tools and Environments for Distance Education. Contribution to VirtUE - a Virtual University for Europe. Disponível em: <http://dreamteam.fernuni-hagen.de/paper/platfrms.pdf>. Acesso em: 20 Jun. 2005.

[ROT2000] Roth, J.; Unger, C. (2000). Developing synchronous collaborative applications with TeamComponents. Proceedings of the 4th International Conference on the Design of Cooperative Systems, p. 23-26.

[SUT2001] Suthers, D. (2001). Architectures for Computer Supported Collaborative Learning. Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT2001), p. 25-28.

[VER1997] Vergnaud, G. (1997). The nature of mathematical concepts. In: T. Nunes e P. Bryant (Eds.), Learning and teaching mathematics: An international Perspective, Psychology Press, Hove, p. 5-28.

Dissertação de Mestrado apresentada por **Enoque Calvino Melo Alves** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **"Design de Componentes Educacionais Síncronos"**, orientada pelo **Prof. Alex Sandro Gomes** e aprovada pela Banca Examinadora formada pelos professores:

Prof. Nelson Souto Rosa
Centro de Informática / UFPE

Profa. Magdala de Araújo Novaes
Núcleo de Tese-Saúde / UFPE

Prof. Alex Sandro Gomes
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 31 de agosto de 2005.

Prof. JAELSON FREIRE BRELAZ DE CASTRO
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.