UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Mecanismo de Suporte à Percepção em Ambientes Cooperativos

por

MANUELE KIRSCH PINHEIRO

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. José Valdeni de Lima
Orientador

Porto Alegre, fevereiro de 2001

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pinheiro, Manuele Kirsch

Mecanismo de suporte à percepção em ambientes cooperativos / por Manuele Kirsch Pinheiro. – Porto Alegre: PPGC da UFRGS, 2000.

167p. : il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2000. Orientador: Lima, José Valdeni de.

1. percepção. 2. groupware. 3. CSCW. I. Lima, José Valdeni de. II. Título

Agradecimentos

“It’s just a moment

This time will pass”

from the music “Stuck in a moment you can’t get out of” by U2

O mestrado representou um período em que muita coisa aconteceu em minha vida, e no qual muitas pessoas participaram e ajudaram. Será meio difícil conseguir colocar todo mundo em uma única página, mas prometo que vou fazer o possível. Se esquecer de alguém, peço desculpas.

A primeira pessoa a quem eu preciso agradecer, e muito, não só pela inestimável ajuda durante esta dissertação, mas também por tudo que representa, é ao meu marido e companheiro, Luiz Angelo. Sem ele, muito dessa dissertação não teria sido possível. Obrigada, amor.

Claro, não poderia esquecer da pessoa que mais paciência teve comigo. Já são doze anos de paciência e amizade. Obrigada, Andrea, por todo o apoio, mesmo quando era você quem mais precisava dele, você estava lá. Nunca vou me esquecer disso.

A minha família, essa eu tenho muito a agradecer. À minha mãe, muito obrigada, não só pela maravilhosa revisão, mas principalmente pelas palavras de apoio e por ter me convencido a ficar em Porto Alegre. Ao meu pai, por ter me aceitado em sua casa e permitido que eu concluísse meus estudos com a calma necessária. À minha irmã, Damaris, por todo o incomparável companheirismo, principalmente enquanto eu estive no Rio de Janeiro, obrigada por tudo (e é claro, pela máquina de lavar também!). Enfim, a toda minha família, meu irmão, cunhada, sobrinha e mascotes, pela paciência comigo e minhas ausências. Obrigada!

E por falar no Rio de Janeiro, eu não posso deixar de agradecer ao Prof. Marcos que permitiu minha ida para lá e, junto com sua equipe, garantiu-me uma experiência profissional e de vida únicas. Muito obrigada a você, Marcos e a toda a equipe: Cláudia Motta, Yoko (e família também!), Marcelo, Jacaré, Renata, Flávia Baiana, Flávia, Cláudia, Zapico, Zero, enfim, toda a turma que me recebeu tão bem e me ensinou um pouco do jeito carioca de ser. Meu obrigada e uma confissão: não consigo mais ir a uma churrascaria sem lembrar do Porcão (isso, sem falar nas praias, porque daí é covardia)!

Enfim, gostaria de agradecer a todos os colegas de mestrado, aos funcionários e professores do Instituto e ao Prof. Valdeni, por toda a ajuda e lições que todos me deixaram. Meu agradecimento carinhoso a todos vocês e também àqueles que esqueci de diretamente mencionar aqui. Faltou espaço na folhinha...

Sumário

Lista de Abreviaturas

API	*Application Programming Interface*
APO	*Appliaction Plus Objects*
BSCW	*Basic Support for Cooperative Work*
BW	*Big Watcher*
CGI	*Common Gateway Interface*
COPSE	*Collaborative Project Support Environment*
CSCW	*Computer Suported Cooperative Work*
CUTE	*Collaborative UML Technique Editor*
EDI	*Eletronic Document Interdate*
ENP	*Event Notification Protocol*
ER	Entidade-Relacionamento
Fig.	FIGURA
GDSS	*Sistemas de suporte a decisão*
HCI	*Human-Computer Interaction*
HTML	*HyperText Markup Language*
HTTP	*HiperText Transfer Protocol*
IETF	*Internet Engeneering Task Force*
JDBC	*Java DataBase Connectivity*
JDK	*Java Development Kit*
LAN	*Local Area Network*
MVC	*Model – View – Controller*
POLIAwaC	*POLITeam Awareness Client*
QoS	*Quality of Service*
RPC	*Multicast Remote Procedure Calls*
SGBD	*Sistema Gerenciador de Banco de Dados*
Tab.	TABELA
TCP/IP	*Transmission Control Protocol / Internet Protocol*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WAP	*Web Awareness Protocol*
WEBDAV	*Distributed Authoring and Versioning on the Web*
WWW	*World Wide Web*
WYSIAWIS	*What You See Is Almost What I See*
WYSIWID	*What You See Is What I Did*
WYSIWIS	*What You See Is What I See*
WYSIWYG	*What You See Is What You Get*
XML	*Extensible Markup Language*

Lista de Tabelas

Lista de Figuras

Resumo

O objetivo deste trabalho é desenvolver um mecanismo para suporte à percepção de eventos no passado. Percepção pode ser conceituada como o conhecimento sobre as atividades do grupo, passadas, presentes e futuras, sobre o próprio grupo e seu *status* geral. Sem este conhecimento, o trabalho cooperativo coordenado e estruturado torna-se quase impossível. O suporte à percepção pode ser dividido em seis questões (o que, quando, como, onde, quem e quanto), analisadas sob ponto de vista de sistemas assíncronos e síncronos. A questão "quando" analisa o momento em que ocorre uma atividade, o que gera um evento, podendo ser no "passado", "passado contínuo", "presente" ou "futuro". Uma atividade no "passado" é aquela que foi concluída em um momento passado e cujo registro interessa às outras atividades. Apesar de sua importância, o suporte à percepção de eventos no passado é ainda muito limitado nas ferramentas de *groupware* hoje disponíveis. Como conseqüência, situações como a ausência de um membro do grupo por um certo período de tempo não são tratadas. Como estas situações de ausência são bastante comuns, durante o trabalho em grupo, o seu tratamento é fundamental em um *groupware.* Desta forma, a ausência de membros do grupo exige a contextualização não apenas daqueles que continuam no trabalho, mas, principalmente, daqueles que retornam ao ambiente cooperativo. Neste trabalho, é apresentado um mecanismo flexível para o suporte à percepção de eventos no passado destinado a cobrir a referida contextualização. Este mecanismo foi construído na forma de um *framework,* projetado para ser flexível a ponto de poder ser incluído em qualquer ferramenta de *groupware,* desde que seu autor o queira. Este *framework*, chamado de BW (*Big Watcher*), foi organizado em quatro pacotes: três independentes, que trocam informações, descritas no quarto pacote, através somente de classes de fachada. Estas informações são essencialmente eventos, os quais representam as atividades realizadas e já concluídas por algum membro desempenhando um papel dentro do grupo. Estas atividades são registradas pelo *groupware* junto ao *framework,* de modo que este *groupware* possa, através do *framework,* contextualizar seus membros. Além disso, o *groupware* também pode especializar várias classes dentro do *framework* BW, como a descrição dos papéis e a própria descrição dos eventos. Assim, este *framework* pode ser integrado a qualquer ferramenta de *groupware* em ambiente assíncrono que necessite de um mecanismo para o suporte à percepção de eventos no passado, para evitar que situações de ausência prejudiquem o andamento dos trabalhos. Finalmente, foi implementado e testado o *framwork* BW sobre o *groupware* CUTE/COPSE para validar as idéias desta dissertação.

Palavras-Chave: percepção, *groupware*, CSCW.

Title: "FRAMEWORK FOR PAST EVENTS AWARENESS SUPPORT"

Abstract

The aim of this work is to develop a support mechanism for past events awareness. Awareness is the knowledge on group activities, including past, presents and future activities. It's also the knowledge about the group itself and its overall status. Without this knowledge, a coordinated and structured cooperative work became almost impossible. The awareness support can be divided in six questions (what, when, where, who, how and how much) over synchronous and asynchronous systems. The "when" question analyses when an activity that raises an event occurs. This event can be either on "past", "continuous past", "present" or "future". An event occurred on "past" is one that had finished on the past and whose record is interesting to the other activities. Despite of its importance, the past events awareness support is very limited on groupware systems available today. Thus, many situations, like when a group member is absent for a while, aren't handled. As these situations are very common into a group work, handling them is a main factor to groupware systems. Because of these situations, the group members' absence demands an awareness support, not only to the members that are still working, but specially to those members that are returning to the cooperative environment. This work presents a flexible mechanism for past events awareness support that covers those situations. This mechanism was built as a framework, designed to be flexible enough to be included in any groupware systems, when the groupware author desires that. This framework, called Big Watcher (BW), is divided in four packages: three independent ones that talk to each other only through facade classes and exchange informations described in the fourth package. These informations are, in essence, events. Those events represent group activities that were already finished by some member playing a role inside the group. Those activities are registered by the groupware in the framework, in such way that the groupware, through this framework, becomes able to give awareness to its members. Beside of that, the groupware may specialize many classes in framework BW, for example, the class that describes papers and the class that describes events. Thus, this framework can be included into any groupware system in an asynchronous environment that needs a past events awareness support to prevents that the group work could be damaged by those absence situations. Finally, the BW framework was implemented and tested into the COPSE/CUTE groupware system, validating the statements presented in this work.

Keywords: awareness, groupware, CSCW.

1 Introdução

Nas décadas de 80 e 90, presenciou-se grandes avanços tecnológicos e um crescente uso destes avanços na busca por maior produtividade e qualidade, visando um melhor posicionamento no mercado. Este mesmo mercado tem se mostrado cada vez mais dinâmico e competitivo e, para ser competitiva e sobreviver neste ambiente, uma empresa precisa ter, de forma consistente, velocidade na tomada de decisões de gerência, no desenvolvimento de produtos e no atendimento ao cliente [CAR 91]. Entre estes avanços tecnológicos, pode-se citar o avanço das redes de computadores e o fenômeno Internet. Sobre este fenômeno, tem-se visto o desenvolvimento de novas ferramentas de comunicação as quais permitem às pessoas trabalharem de forma mais flexível [NOM 98]. Através destas novas ferramentas, torna-se possível satisfazer melhor restrições econômicas e gerenciais dos usuários e uma dessas novas áreas de aplicações é o Trabalho Cooperativo Suportado por Computador _ *Computer Supported Cooperative Work* (CSCW) [KLO 99].

A área de CSCW surgiu exatamente destes avanços tecnológicos e principalmente da constante busca por maior produtividade e qualidade. Esta busca vem incentivando o uso de equipes de trabalho para a solução de problemas cada vez mais complexos com a qualidade e produtividade desejadas, ponto da partida para a área de CSCW. Esta objetiva, principalmente, estudar o trabalho em grupo e meios de apoiá-lo sob vários aspectos, com a intenção de aumentar a produtividade do grupo. Com isso, questões do tipo: como as pessoas trabalham juntas em grupo e do que elas precisam para isso, e como os computadores e suas ferramentas podem ser desenvolvidos para suportar estas pessoas e as atividades nas quais estão engajadas, são preocupações de CSCW [CSC 98].

Estas preocupações levaram a área de CSCW, entre outras coisas, a valorizar o conceito de percepção (ou *awareness*). A percepção pode ser conceituada como a contextualização das atividades individuais através da compreensão das atividades realizadas pelos demais e seu suporte permite transformar interações irregulares em interações consistentes e perceptíveis no decorrer do tempo, possibilitando que os membros se mantenham atualizados sobre importantes eventos dentro do grupo [DIA 98],[ARA 97]. Sem um suporte adequado à percepção, o trabalho em grupo pode se tornar confuso e truncado, já que os membros não têm conhecimento sobre aquilo que aconteceu e vem acontecendo dentro do grupo, enfim não possuem noção do contexto onde estão inseridas suas próprias atividades e contribuições. O não conhecimento deste contexto permite que surjam contribuições conflitantes, redundâncias e conflitos dentro do trabalho em grupo, o que conduz a um produto final de baixa qualidade e produtividade questionável, resultado totalmente contrário aos objetivos esperados do trabalho em equipe.

Todavia, apesar da importância da percepção para o sucesso do trabalho em equipe e do crescente interesse que os pesquisadores da área de CSCW têm depositado, como mostra o crescente número de *workshops* específicos de *awareness*, como o ocorrido junto ao CHI'97 (ver Brinks e McDaniel [BRI 97]), o mesmo entusiasmo com o assunto não tem sido observado em muitas ferramentas de *groupware*. Muitas destas não vêm apresentando um suporte adequado à percepção, deixando, dessa forma, uma

lacuna no suporte ao trabalho em grupo. A situação é ainda pior quando o assunto é o suporte a acontecimentos passados dentro do grupo. Este tipo de suporte, muito necessário principalmente em ambiente de trabalho assíncrono, consiste em fornecer aos participantes do grupo informações sobre as atividades passadas, sobre aquilo que já foi feito e concluído dentro do grupo. Este suporte é praticamente ausente dentro da grande maioria de ferramentas de *groupware*, mas sua importância é inegável à medida que decisões, fatos e atividades passadas vão influenciando as presentes. A ausência de suporte à percepção destes acontecimentos passados representa a ausência da manutenção de um histórico do que já foi feito dentro do grupo e isto pode conduzi-lo a repetir erros e decisões equivocadas cometidas no passado.

Este suporte à percepção de acontecimentos passados, que já ocorreram dentro do grupo, consiste no foco central deste trabalho, onde o objetivo é desenvolver um mecanismo flexível que faça o monitoramento destas atividades passadas realizadas dentro do grupo e a posterior contextualização dos participantes dentro de ferramentas de *groupware* quanto a estas atividades. Este mecanismo, construído na forma de um *framework*, foi projetado para ser adaptado e introduzido dentro de ferramentas de *groupware*, fornecendo-lhes o suporte a esta percepção de acontecimentos passados que antes lhes era ausente, a um custo menor do que seria a reconstrução destas ferramentas com a finalidade de desenvolver e introduzir este suporte.

Para mostrar o problema abordado e todo o desenvolvimento desta solução proposta, este trabalho organiza-se nos seguintes capítulos: inicialmente é feita uma revisão geral sobre a área de CSCW no Capítulo 2; em seguida o Capítulo 3 trata em específico da questão da percepção, apresentando uma classificação para esta questão com base nas principais bibliografias sobre o assunto disponíveis; no Capítulo 4, especifica-se o problema tratado neste trabalho e apresenta-se uma visão geral da solução proposta para, a partir do Capítulo 5, introduzir a descrição detalhada desta solução; esta começa pela descrição da arquitetura e do modelo de dados adotado pelo mecanismo dentro do Capítulo 5 e segue no Capítulo 6 com a descrição do *framework* desenvolvido, com destaque especial para o mecanismo de eventos adotado e ainda o guia de interfaces, enquanto o Capítulo 7 descreve o protótipo desenvolvido com o *framework* proposto. Por fim, são apresentadas as conclusões e trabalhos futuros para esta dissertação no Capítulo 8 (conclusão) e a bibliografia consultada. Convém mencionar aqui que os capítulos 3 a 7 apresentam uma organização bem definida: inicialmente é feita uma rápida introdução sobre o capítulo e sua importância para o conjunto do trabalho, seguida pelo texto do capítulo propriamente dito e encerrando com uma seção de considerações finais, onde se relacionam as contribuições daquele capítulo para o trabalho e os próximos passos na seqüência.

2 Trabalho Cooperativo Suportado Por Computador

A sociedade globalizada de hoje vem exigindo das empresas resultados rápidos e eficazes, incentivando a competição e o livre mercado. Esta realidade está obrigando muitas empresas a repensarem sua forma de produção e seu uso da tecnologia, a fim de se adaptarem a este novo mercado. Isto também determinou uma mudança no ambiente de produção, que deixou de ser seqüencial e seguidamente fragmentado, como na década de 80, cedendo lugar a um ambiente dinâmico, onde as informações precisam trafegar e produzir resultados rapidamente. Hoje, ser competitivo requer, consistentemente, maior velocidade na tomada de decisões de gerência, no desenvolvimento de produtos e no atendimento ao cliente [CAR 91]. Esta procura por uma maior eficiência na solução de problemas cada vez mais complexos tem feito com que atividades, antes individuais, passassem a ser resolvidas agora em grupos de trabalho [SPU 94], [PIN 99a]. Além disso, fatores como a disseminação das redes de computadores e dos sistemas distribuídos, a distribuição das organizações e a necessidade de compartilhar informações e recursos têm incentivado cada vez mais a formação de grupos de trabalho multidisciplinares e geralmente distribuídos [SOU 96], [FER 98], [HOL 94].

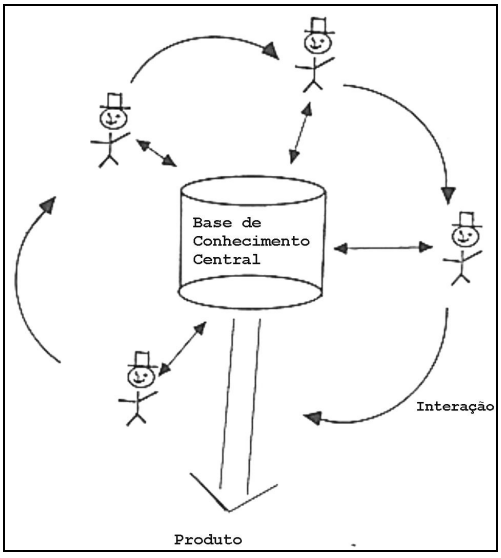


FIGURA 2.1 - Trabalho em grupo interativo [SPU 94]

O objetivo na formação destas equipes é fazer com que os membros possam cooperar entre si, cada um contribuindo com sua parte, para chegarem ao produto final [SPU 94]. A Fig. 2.1 acima, baseada em [SPU 94], demonstra bem esta idéia. Na figura, os membros de um grupo interagem entre si e depositam suas contribuições em uma "base de conhecimento" comum, uma espécie de "memória do grupo". Eles trocam idéias, negociam soluções e discutem contribuições e tudo isso leva a mais contribuições depositadas na memória do grupo. Como resultado deste processo, dessa base de conhecimento formada a partir das contribuições e das interações entre os membros, tem-se o produto final. Este produto final não se resume ao mero somatório

das contribuições dos membros, mas se trata de um conjunto que expressa as opiniões do grupo como um todo. Não são as idéias de um ou dois membros impostas aos demais, mas um conjunto consistente e coerente, resultado direto das interações entre os membros, formando uma entidade única, o grupo, mais poderosa e eficiente do que a simples reunião destes mesmos membros isolados.

É nesse ambiente que a área de Trabalho Cooperativo Suportado por Computador (ou CSCW - *Computer Supported Cooperative Work*) tem se destacado, uma vez que seus objetivos são justamente os de estudar o trabalho em grupo e meios de apoiá-lo sob vários aspectos, com a intenção de aumentar a produtividade do grupo. Segundo Ellis et al. [ELL 91] “CSCW observa como os grupos trabalham e procura descobrir como a tecnologia, especialmente computadores, pode ajudá-los a trabalhar” e um *groupware* é a classe de aplicações para pequenos grupos e organizações, que surge da união dos computadores com as grandes bases de informações e a tecnologia de comunicação. *Groupware* são ferramentas projetadas para o suporte computadorizado a este trabalho em equipe e seu objetivo é apoiar e incentivar o grupo, para que este possa cooperar e alcançar os resultados esperados desta cooperação.

Dessa forma, o próprio mercado e sua necessidade por eficiência e resultados rápidos tem impulsionado o desenvolvimento de ferramentas de *groupware*, fazendo com que soluções baseadas nestas estejam emergindo para fornecer maior competitividade às empresas [SOU 96]. Com isso, desde a década de 80, quando o termo CSCW foi cunhado, tem-se presenciado um crescente interesse nesta área, que também tem se mostrado multifacetada, envolvendo o estudo de tecnologias e de aspectos sociais e cognitivos. Como conseqüência deste crescente interesse, percebe-se também um crescimento na quantidade de ferramentas de *groupware*, as quais representam a tecnologia gerada pelas pesquisas em CSCW [BOR 95], [SAL 98], [SIE 94]. Este crescimento deu-se tanto em âmbito de pesquisas, quanto em âmbito comercial. Em âmbito de pesquisas, há uma consciência generalizada entre seus profissionais de que a atividade cooperativa é uma das principais formas de fazer frente às necessidades de soluções rápidas para problemas complexos [DIE 96], destacando-se diversas ferramentas desenvolvidas neste meio como CUTE [DIA 97], Byzance [OPE 99], GroupKit [GRE 00], BSCW [GMD 00b], Grove [ELL 91], CoWeb [JAC 96], MUT & OCE [FRI 97], Habanero [CHA 98], entre muitas outras (ver exemplos em [PIN 99a], [SAL 98], [VES 98], [ANU 94]). Em âmbito comercial, o crescente número de produtos como Lotus Notes, Microsoft NetMeeting e CoolTalk do Netscape, segundo Prakash et al. [PRA 99], evidencia a popularidade da tecnologia de CSCW.

Todas estas ferramentas objetivam “aumentar o potencial do grupo, fazendo com que o resultado seja maior que a soma das contribuições individuais de cada membro do grupo” [DIA 98]. Todavia, chegar a este estado sinérgico no grupo não consiste em uma tarefa fácil. A definição de cooperação como "trabalhar ou agir em conjunto para um objetivo comum" não captura diversos aspectos desse processo [BEL 95]. O mero trabalho envolvendo várias pessoas não caracteriza necessariamente uma cooperação, já que a cooperação surge quando todos se comprometem com um objetivo concreto comum. Logo, um trabalho envolvendo competição entre as partes, apesar da presença da interação entre os membros, não se classifica como cooperativo. Na verdade, o que se espera do trabalho em grupo é o desenvolvimento de produtos com qualidade e produtividade, que podem ser obtidas pela interação harmônica e realmente cooperativa entre os membros, fazendo com que estes produtos revelem as idéias do grupo como um todo [DIA 98], [DIE 96].

Esta desejada produtividade depende de que o entendimento entre os participantes seja alcançado da forma mais organizada, breve e eficiente possível. Este entendimento está baseado nos conceitos de incerteza e "equivocabilidade" e aos quais se deseja diminuir. O primeiro, incerteza, refere-se à ausência de informação e o segundo refere-se à ambigüidade, à existência de interpretações conflitantes sobre os assuntos tratados pelo grupo. Assim, o trabalho em grupo pode ser visualizado como um processo contínuo de redução dos graus de incerteza e equivocabilidade, no qual a solução de conflitos e a convergência nos pontos de vista determinam a qualidade dos resultados [ARA 97], [DIA 98].

Para viabilizar este entendimento entre os participantes, Araújo et al. [ARA 97] apresentam quatro questões: a comunicação entre os envolvidos, a coordenação das atividades, a memória do grupo, que é o registro do conhecimento comum ao grupo, e a percepção do grupo com relação ao contexto do trabalho (este último aspecto é central nesta dissertação e encontra-se discutido em detalhes no próximo capítulo). A coordenação refere-se ao gerenciamento e ao acompanhamento das atividades realizadas pelo grupo e individualmente por cada participante. A comunicação entre os membros reside na existência de ligações entre eles, através tanto de canais de comunicação direta, como troca de mensagens e reuniões, quanto via um canal indireto através da memória de grupo, onde a construção e o compartilhamento do conhecimento comum podem ser considerados interfaces de comunicação. Já a memória de grupo propriamente dita registra todo o processo de interação do grupo, como a própria comunicação realizada e passos desencadeados, bem como todos os produtos gerados por esta cooperação. A percepção, que será discutida em detalhes no próximo capítulo, é definida como a contextualização das atividades individuais através da compreensão das atividades realizadas pelos demais e seu suporte permite transformar interações irregulares em interações consistentes e perceptíveis no decorrer do tempo, permitindo que os membros se mantenham atualizados sobre importantes eventos dentro do grupo [DIA 98],[ARA 97]. A Fig. 2.2 abaixo esquematiza estes aspectos apontados por Araújo et al. os quais podem ser vistos como resultados da própria natureza da cooperação, intimamente ligada aos processos básicos de comunicação, coordenação, negociação, co-realização e compartilhamento, discutidos a seguir.

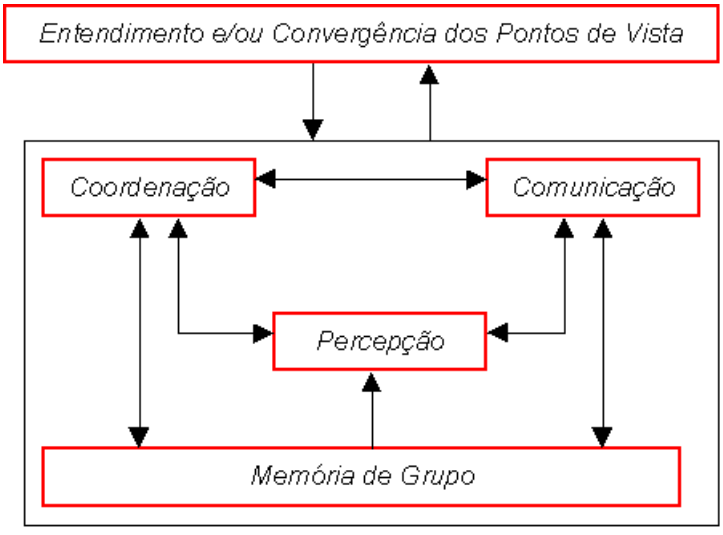


FIGURA 2.2 - Aspectos de suporte por computador à cooperação [DIA 98]

A colaboração efetiva necessita que as pessoas compartilhem informações. Este compartilhamento está intimamente ligado à comunicação entre os membros, a qual pode ser melhorada se as atividades do grupo são coordenadas [ELL 91]. A comunicação é fundamental, sendo necessário aos participantes dispor de um mecanismo com esta finalidade e uma linguagem comum compreendida por todos. Já para a coordenação é vital para garantir a eficiência da colaboração, pois sem ela os participantes podem se envolver em tarefas conflitantes e repetitivas. Logo, para o processo de coordenação é preciso identificar o suporte necessário, com mecanismos de controle sobre a própria comunicação e de monitoramento, para evitar a dispersão do grupo, entre outras coisas [RAP 99],[BOR 95].

Além dos processos de comunicação e de coordenação, a cooperação também envolve naturalmente os processos de negociação, co-realização e compartilhamento. A negociação é necessária quando discussões são geradas e um consenso é preciso, e a co-realização, assim como o compartilhamento, estão presentes na cooperação à medida que esta envolve o desenvolvimento de um produto ou objeto compartilhado que é manipulado por todos os membros [BOR 95]. A Fig. 2.3 abaixo esquematiza estes processos presentes na cooperação.

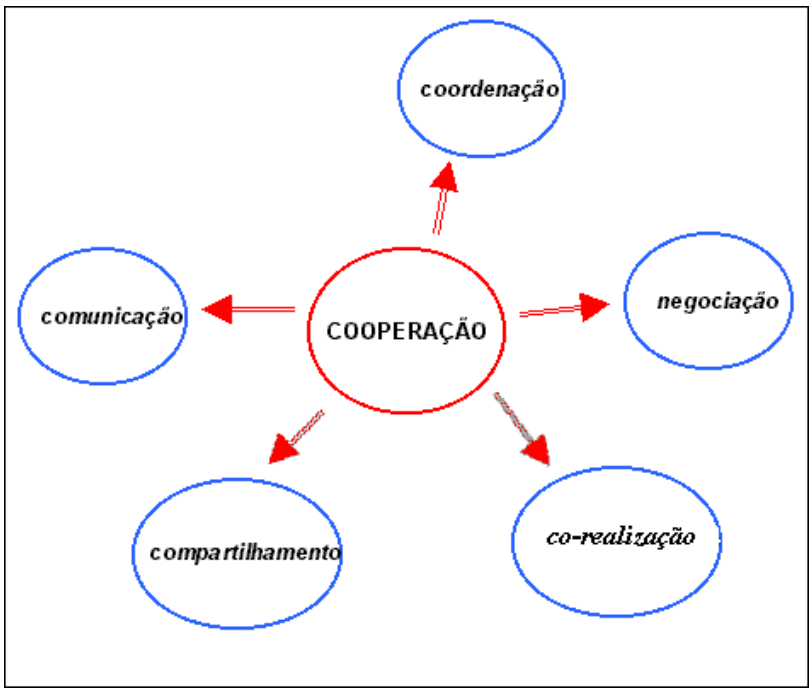


FIGURA 2.3 - Processos envolvidos na cooperação

Portanto, para suportar a cooperação entre um grupo de pessoas de forma adequada, uma ferramenta de *groupware* precisa abranger os processos apresentados na Fig. 2.3, através essencialmente do suporte aos quatro aspectos (coordenação, comunicação, memória de grupo e percepção) apresentados na Fig. 2.2. Dessa forma, surge uma série de requisitos necessários para que um *groupware* obtenha o sucesso esperado em seu objetivo de auxiliar e apoiar o trabalho em grupo. Estes requisitos incluem ser confiável e dar um correto suporte à informação compartilhada; facilitar a cooperação e não impor práticas que causem mudanças radicais na forma de trabalho do grupo; não manter as informações usadas no trabalho cooperativo sob o domínio de

somente um indivíduo; ser composto preferencialmente por aplicações menores e interrelacionadas e suportar ambientes heterogêneos e abertos; reconhecer mudanças no contexto e permitir a redefinição de processos e procedimentos, além de fornecer percepção da interação do grupo [BOR 95], [DIE 96].

Estes são alguns exemplos de requisitos que podem ser apontados como importantes para o sucesso de um *groupware*. Cada um destes procura, direta ou indiretamente, dar suporte a algum dos processos apresentados na Fig. 2.3, facilitando assim a cooperação como um todo. Todavia, além de se observar os fatores de sucesso de um *groupware*, também existem fatores que podem ser decisivos para o seu insucesso ou a sua não aceitação pelo seu grupo de usuários considerado alvo. Alguns destes fatores são: a disparidade entre quem se beneficiará com a aplicação (*groupware*) e quem deverá realizar o trabalho adicional para suportá-la; a tendência entre os gerentes em favorecer aplicações que os favoreçam; e a dificuldade de avaliar custos e benefícios destas aplicações [BOR 95].Outros fatores que podem levar um *groupware* a falhar podem ser encontrados em [GRU 98].

Todos estes requisitos para o sucesso e fatores de insucesso são gerais aos mais diversos tipos de *groupware* que podem ser encontrados. Tratando-se de tipos, as ferramentas de *groupware* podem ser classificadas segundo diversas taxinomias, algumas das quais são apresentadas na próxima seção.

2.1 Classificações em CSCW

Desde o princípio das pesquisas em CSCW, diversas classificações para suas aplicações foram propostas por inúmeros autores. Estas classificações levam em conta diferentes critérios para análise, fazendo com que cada uma veja as pesquisas em CSCW sob sua ótica, completando-se de modo a formar um quadro mais amplo e descritivo da área. As principais classificações encontradas na bibliografia são apresentadas nesta seção.

2.1.1 Tempo x Espaço

A classificação tempo X espaço apresentada por Ellis et al. [ELL 91] é, sem dúvida alguma, a mais aceita e conhecida entre todas as classificações. Ellis et al. [ELL 91] propõem que um *groupware* pode ser concebido tanto para ajudar um grupo face a face quanto distribuído, ou para aprimorar a comunicação e a colaboração dentro de uma interação em tempo real ou não. Estas considerações de espaço e tempo sugerem a representação de quatro categorias de *groupware*, apresentadas nas Fig. 2.4.

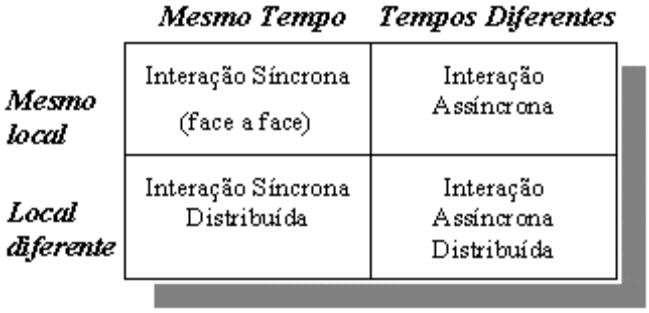


FIGURA 2.4 - Matriz Espaço x Tempo proposta por Ellis et al. [ELL 91]

A classificação apresentada na Fig. 2.4 divide os *groupware* em duas dimensões, uma para o espaço, tratando da localização física dos participantes, e outra temporal, tratando do momento em que os participantes trabalham, a qual se divide em mesmo tempo (síncrona) e tempos diferentes (assíncrona). Para Willians et al. [WIL 94], em uma interação síncrona, a presença dos usuários cooperantes é requerida, o que não ocorre na interação assíncrona, onde os usuários trabalham em diferentes momentos no tempo. Já para Salcedo [SAL 98], o que define uma interação assíncrona é a presença de uma defasagem entre uma ação e sua percepção pelos demais autores. Em resumo, tem-se tipicamente, em ambientes síncronos, usuários interagindo simultaneamente, sob um mesmo conjunto de dados, através de um espaço de informações compartilhados, enquanto que, em ambientes assíncronos, tem-se os membros do grupo trabalhando em diferentes momentos, mas ainda interagindo sobre um mesmo conjunto de dados.

FIGURA 2.5 - Exemplos de sistemas de *groupware* [SAL 98]

Sobre as quatro categorias definidas na Fig. 2.4, pode-se distribuir exemplos de *groupwares*, de acordo com suas características. A Fig. 2.5 acima traz alguns destes exemplos. A categoria “Local diferente” x “Mesmo tempo”, por exemplo, inclui aplicações como Vídeo-Conferência e Ensino à Distância, considerados complexos. Essa complexidade decorre não só do uso de interação síncrona, mas também da distribuição geográfica dos participantes. Esta distribuição remete às questões de compartilhamento de recursos distribuídos, envolvendo tecnologias como Banco de Dados, Sistemas Distribuídos, Redes de Computadores e Sistemas Operacionais. Além disso, um *groupware* compreensivo poderá adequar melhor necessidades em todos os quadrantes, como é o caso da presença da edição cooperativa, a qual pode ser encaixada em todas as quatro categorias, conforme pode ser observado na Fig. 2.5. Segundo Salcedo [SAL 98], os sistemas de edição cooperativa cobrem todas as categorias, porque existem em diversas situações, baseadas nas necessidades dos autores e do campo de aplicação dos documentos gerados.

2.1.2 Tempo x Espaço x Previsibilidade

Outra classificação para sistemas de *groupware* é a apresentada por Grudin (apud Dias [DIA 98]), onde o autor propõe a inclusão do critério de previsibilidade à matriz espaço x tempo apresentada anteriormente. Grudin divide a coluna "tempos diferentes" em "tempo diferente, mas previsível", indicando que se pode prever o momento em que os participantes estarão ativos, e "tempo diferente e imprevisível", quando não é possível fazer esta previsão, e divide a coluna "local diferente" em "local diferente, mas previsível", para quando é possível prever o local onde estará cada participante e "local diferente e imprevisível", para quando não é possível se afirmar qual será a localização física dos participantes. A Fig. 2.6 abaixo traz alguns exemplos de *groupware* classificados segundo esta taxinomia, retirados de Dias [DIA 98].

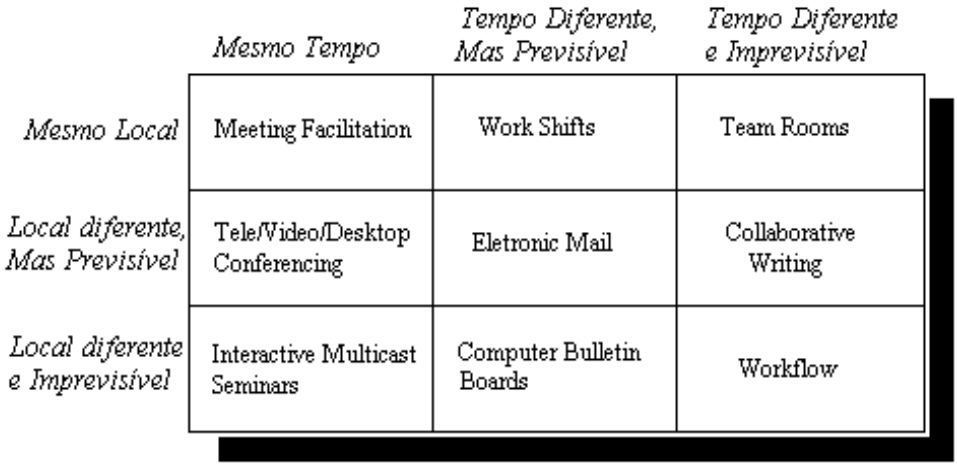


FIGURA 2.6 - Classificação Tempo - Espaço – Previsibilidade [DIA 98]

2.1.3 Tempo x Espaço x Tamanho do Grupo

Uma terceira classificação para CSCW que envolve as dimensões de espaço e tempo é a proposta por Nunamaker (apud Dias [DIA 98]). Esta classificação propõe uma matriz de três dimensões, mostrada na Fig. 2.7 abaixo. A primeira dimensão, tempo, é dividida em mesmo tempo e tempos diferentes, como na classificação proposta por Ellis et alli. A segunda dimensão, espaço ou proximidade do grupo, é dividida em indivíduos distribuídos, grupos distribuídos e indivíduos distribuídos em grupos, e a última dimensão, tamanho do grupo, é dividida em "de 3 a 7 pessoas" e "de 7 a n pessoas".

A maior peculiaridade desta classificação reside justamente na definição desta terceira dimensão, o tamanho do grupo. Esta definição deve ser considerada durante o desenvolvimento de ferramentas de *groupware*, pois o tamanho do grupo vai determinar uma série de cuidados para que o suporte à comunicação e ao compartilhamento de informações seja adequado, mesmo para grandes grupos dispersos. Outra peculiaridade é quanto à dimensão espaço, que é vista como o grau de proximidade do grupo e não aborda grupos face a face, como nas classificações anteriores.

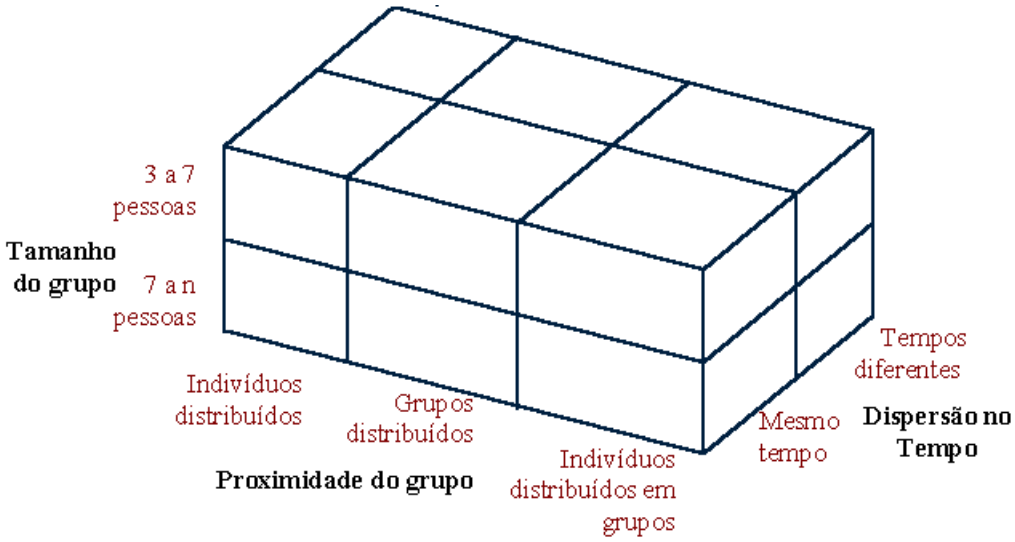


FIGURA 2.7- Classificação Tempo - Espaço - Tamanho do grupo [DIA 98]

2.1.4 Classificação quanto à funcionalidade

Ellis et al. [ELL 91] propõem ainda uma segunda classificação, levando em conta a funcionalidade à nível de aplicação. Nesta classificação, os sistemas de *groupware* são divididos em diversas categorias, de acordo com suas funções, podendo também haver sobreposição entre as categorias, pois com o aumento da demanda por sistemas integrados, vê-se uma maior mistura nestas funcionalidades [ELL 91]. Estas funcionalidades são as seguintes:

- *Sistemas de Mensagens*: para Ellis et al. este é o tipo mais familiar de *groupware*, suportando a troca assíncrona de mensagens textuais entre grupos de usuários. Os mais clássicos exemplos desta categoria de *gropuware* são o correio eletrônico e as BBS [ELL 91].
- *Editores Multiusuários*: um grupo pode fazer uso de editores multiusuários para compor e editar em conjunto um documento [ELL 91]. Segundo Salcedo [SAL 98], um conjunto de autores pode produzir resultados melhores e de modo mais eficiente se combinarem suas habilidades, através do confronto de suas experiências próprias, do seu nível de especialização e das contribuições de cada um. Tipicamente, estes sistemas dividem o objeto que está sendo editado em segmentos lógicos e permitem o acesso concorrente para a leitura a qualquer segmento, mas a escrita é limitada a uma única permissão por segmento [ELL 91]. Estes sistemas também fazem, de forma transparente aos usuários, o controle de acesso e a sincronização entre os segmentos e somente alguns exemplos destes adotam um modelo de notificação, como o BSCW [BEN 98], [GMD 00a], ou um esquema de controle de versões, como o PREP [SOU 96], [DIE 96], ao invés de um controle de acesso mais rígido.

- *Sistemas de Suporte à Decisão do Grupo (Group Decision Support Systems _ GDSS) e Salas de Reunião Eletrônicas*: os GDSS empregam os computadores para a exploração de problemas não estruturados em um grupo e junto com as salas de reunião eletrônicas objetivam aumentar a produtividade de reuniões para a tomada de decisões, através da aceleração deste processo ou do aumento na qualidade da decisão tomada [ELL 91].
- *Conferências*: existem vários tipos de suporte computadorizado a conferências, sendo que Ellis et al. [ELL 91] citam três destes tipos. O primeiro, as conferências em tempo real, permitem a um grupo, ligados em uma sala de reunião eletrônica ou fisicamente dispersos, interagir sincronamente; o segundo, teleconferências, suportam a interação do grupo através de telecomunicações; e a terceira, *desktop conferencing*, usam as estações de trabalho como interface da conferência, mas também rodam outras aplicações compartilhadas entre todos os participantes. Alguns exemplos de aplicações de conferência podem ser encontrados em Claveleira [CLA 98].
- *Agentes Inteligentes*: Ellis et al. [ELL 91] propõem agentes inteligentes como responsáveis por conjuntos específicos de tarefas, vistas pela interface dos usuários como ações semelhantes a dos outros usuários. O efetivo uso destes agentes em sistemas de CSCW tem aparecido principalmente na última década. Um exemplo disso pode ser encontrado em Schmidt et al. [SCH 98], onde os participantes são representados no sistema por um conjunto de agentes, que executam (ou antecipam) para o usuário várias tarefas. A realização de tarefas que seriam tediosas ao elemento humano, mas que são importantes para a atividade cooperativa, como a criação de boletins informativos e a notificação da presença de participantes têm sido a principal utilização de agentes em ferramentas de *groupware* [DIE 96].
- *Sistemas de Coordenação*: tipicamente estes sistemas permitem aos indivíduos ver suas ações e as ações relevantes de seus colegas. Eles podem também disparar ações, informando o estado atual e condições de espera destas, ou ainda gerar limites e alertas automáticos. Alguns modelos de sistemas de coordenação são sistemas orientados a formulários, com foco no roteamento de documentos; sistemas orientados a procedimentos, que vêem os procedimentos organizacionais como processos programáveis; sistemas orientados à conversação, baseados na observação de que pessoas coordenam suas atividades através da comunicação; e modelos orientados a estruturas, os quais descrevem as atividades organizacionais em termos de relacionamentos de papéis [ELL 91].

Esta não é a única classificação que abrange o critério de funcionalidade para as ferramentas de groupware, entretanto é, sem dúvida alguma, uma das mais completas. Vale talvez citar classificações, como a de Grudin [GRU 96], que aborda com maior ênfase os sistemas de *workflow*. Para Grudin [GRU 96], sistemas de gerência de *workflow* modelam seqüências de subtarefas no processo de trabalho e as regras desempenhadas por cada indivíduo. Quando uma subtarefa é concluída, o trabalho é automaticamente roteado para a pessoa responsável pela próxima subtarefa.

2.1.5 Classificação quanto ao tipo de interface

A interface, no domínio de CSCW, é de extrema importância para o sucesso de um *groupware*, uma vez que ela é a responsável pela apresentação das informações compartilhadas para os membros do grupo e também representa o principal canal de acesso entre os participantes. Por estes motivos, o projeto de uma interface dentro de um ambiente cooperativo envolve outros aspectos além daqueles normalmente considerados no projeto de uma interface mono-usuário. O uso por múltiplos usuários pode produzir um nível maior de atividades e um maior grau de concorrência que um único usuário, e a interface deve ser capaz de suportar este comportamento complexo [ELL 91]. Além disso, o próprio *groupware* deve agir como a porta de acesso de um usuário a seus colegas. Esta é apontada por Wilson et al. [WIL 94] como sendo a maior diferença entre um *groupware* e um *software* qualquer: ao contrário dos outros *softwares*, que procuram esconder um usuário dos demais, um *groupware* deve acentuar o ambiente multiusuário, coordenando e orquestrando as atividades, de modo que os usuários possam ver um ao outro e resolver os possíveis conflitos, como um grupo coeso.

Em virtude desta acentuada importância para o sucesso de um *groupware*, o modelo de interface adotado passou a ser considerado um critério de classificação e análise em CSCW. Dias [DIA 98] faz uma compilação de vários autores e apresenta quatro modelos de interfaces possíveis, descritos abaixo:

- *WYGIWIG (What You Get Is What I Get)*: neste modelo de interface, os usuários compartilham janelas e pequenas inconsistências chegam a ser toleradas pelo sistema, desde que não comprometam as informações que podem ser recuperadas do espaço compartilhado pelos usuários. Há também a presença de restrições de coordenação e atrasos na propagação das atualizações também são tolerados.
- *WYSIWID (What You See Is What I Did):* neste modelo, as interações entre os usuários se dão de forma semi-síncrona, com a propagação das alterações para os demais usuários, podendo ser adiada até que condições preestabelecidas sejam satisfeitas, fazendo com que a visão que determinado usuário tenha das informações compartilhadas seja, por um curto espaço de tempo, defasada em relação à real situação atual.
- *WYSIAWIS (What You See Is Almost What I See):* neste tipo de interfaces, os usuários compartilham janelas com variações que não chegam a prejudicar a interpretação dos resultados das ações dos colegas.
- *WYSIWIS (What You See Is What I See):* nas interfaces WYSIWIS, o espaço de trabalho, que é compartilhado entre os membros do grupo, aparece igual para todos os participantes. Assim, se um membro se reposiciona em sua janela, o mesmo ocorrerá na janela de seus colegas. Um exemplo de *groupware* que adota este tipo de interface é o CoWeb [JAC 96]. Este permite que seus usuários adotem um modelo WYSIWIS durante o preenchimento de formulários eletrônicos na Web, permitindo que todos vejam a mesma parte do documento, vendo, por exemplo, quando um colega altera um *text field*. Uma das vantagens desta abordagem é um forte senso contexto compartilhado [ELL 91], contudo esta abordagem pode ser considerada muito inflexível, pois normalmente os usuários querem ter o controle sobre detalhes como o seu

posicionamento nas janelas. Para resolver este problema, foram propostas as interfaces WYSIWIS relaxadas, onde os membros não precisam necessariamente ver as mesmas informações da mesma maneira. Segundo Dias [DIA 98], este relaxamento pode se dar em quatro dimensões: o espaço de apresentação (sobre que objetos deve-se manter o enfoque), tempo de apresentação (com que periodicidade devem ser feitas as atualizações), população do subgrupo (qual o conjunto de usuários que estará sincronizado) e congruência da visão.

Outros autores apontam outras classificações também considerando fatores de interface como critérios. Por exemplo, dos autores Salcedo [SAL 98] e Dietrich [DIE 96] é possível extrair uma classificação em três categorias, voltada especialmente para a edição cooperativa de documentos usando como critério o chamado "acoplamento de Interfaces", que é a forma como as informações compartilhadas são apresentadas na interface:

- *Interfaces Fortemente Acopladas ou WYSIWIS*: Aqui Salcedo [SAL 98] considera que as contribuições de cada autor são integradas ao documento em tempo real, enquanto Dietrich [DIE 96] assume que cada usuário tem informado na sua tela as mesmas informações, apresentadas da mesma forma, que na dos demais, e ambos concordam que as alterações feitas por um autor são repassadas automaticamente, de forma síncrona para os demais.

- *Interfaces de Acoplamento Médio:* os dados apresentados para os usuários são os mesmos, mas a forma com são mostrados pode ser diferente entre os autores [DIE 96];

- *Interfaces de Acoplamento Fraco*: qualquer membro da equipe trabalha a nível de objeto, sem perceber a ação dos demais [DIE 96] ou percebendo as ações dos colegas com certa defasagem, que pode depender da vontade de cada autor em divulgar suas alterações ou da vontade dos demais em recebê-las [SAL 98].

O uso de mecanismos de distribuição, principalmente nas WYSIWIS e de acoplamento médio, tem como principal vantagem o fato de facilitar a comunicação entre os autores e a percepção do grupo por parte destes. Entretanto, o uso de mecanismos síncronos traz algumas desvantagens, como a questão do desempenho, pois em sistemas WYSIWIS, por exemplo, todas as modificações feitas localmente devem ser transmitidas imediatamente às outras instâncias, o que pode causar uma sobrecarga na rede, um aumento no consumo de tempo pela tarefa e, por conseqüência, uma degradação no desempenho [DIE 96]. Para Salcedo [SAL 98] é desejável que um sistema de autoria cooperativa seja flexível, suportando vários modos de interação e interfaces, a fim de se adaptar às circunstâncias e às necessidades dos co-autores. Esta afirmação pode também ser estendida para outros tipos de sistemas, que não somente os de edição cooperativa.

2.2 Tecnologias e Aspectos Sociais em CSCW

Vistas algumas das mais conhecidas classificações para sistemas em CSCW, é possível agora perceber quão complexa e multifacetada esta área do conhecimento tem se mostrado. Os estudos e pesquisas em CSCW têm envolvido não só diversas tecnologias, como também muitos aspectos sociais, oriundos de ciências como a sociologia e a psicologia. O estudo destas tecnologias é decisivo na escolha dos meios e métodos mais propícios para o suporte computadorizado ao trabalho em grupo, enquanto que os aspectos sociais precisam ser estudados, uma vez que a essência deste trabalho reside justamente na interação humana, a qual, por sua natureza, é muito complexa e repleta de nuanças, que podem ser decisivas para a aceitação ou a recusa de uma ferramenta de *groupware*. Esta seção discutirá algumas destas tecnologias utilizadas em sistemas de *groupware* e os principais aspectos sociais aqui considerados.

2.2.1 Aspectos sociais

Os aspectos sociais têm grande destaque no desenvolvimento de um *groupware*, pois a essência deste tipo de sistema é o elemento humano. O grupo deve se sentir à vontade com o sistema e não tê-lo imposto, sob o risco de não ser usado a contento, causando frustrações e perda de produtividade [DIE 96]. Várias são as condições que podem fazer com que o grupo trabalhe melhor. Uma identificação perfeita e significativa de uma tarefa; uma certa autonomia para o grupo e um *feedback* sobre o trabalho são alguns exemplos destas condições. Outros aspectos mais ligados ao relacionamento humano também devem ser levados em consideração durante o desenvolvimento de um *groupware*, como hierarquias, conflitos, não cooperação e resposta a compromissos [BOR 95].

Existem dois aspectos sociais que recebem mais destaque em CSCW, os quais não podem ser negligenciados durante o projeto de um projeto de um *groupware*, sob pena de não tê-lo bem aceito por seu grupo de usuários. São eles a organização hierárquica e os modelos cognitivos.

A organização hierárquica relaciona-se com os papéis desempenhados por cada participante do grupo. Observando-se o cotidiano de empresas, é comum ver membros com funções diferenciadas dentro do grupo, como “gerente” e “chefe de seção”, denotando uma organização hierárquica dentro do grupo. Esta organização pode seguir diversos modelos, com vários níveis. Pode ser totalmente plana, com todos os membros gozando dos mesmos direitos e deveres; pode ser plana, mas com um moderador, responsável por unir todas as contribuições dos demais membros; ou ainda pode ser realmente hierárquica, com um ou mais camadas (gerente, chefe, subchefe) ou com um ou mais papéis distribuídos entre os membros do grupo (“redator”, “leitor”, “coordenador”, por exemplo). Um papel é o conjunto de privilégios e responsabilidades atribuídos a uma pessoa [ELL 91]. Eles especificam as responsabilidades de cada um e a relação entre o indivíduo e o objeto de trabalho [SAL 98].

Cada grupo pode adotar um modelo diferente de organização hierárquica, adaptada as suas necessidades. Segundo Baecker *et al*. [BAE 93], esta escolha de papéis pode depender de vários fatores, incluindo a estrutura organizacional, restrições de

tempo dos participantes e habilidades e conhecimentos dos contribuintes. Todavia, independente da escolha feita, a organização utilizada pelo grupo é peça importante na sua dinâmica, uma vez que seus membros, antes mesmo da chegada do *groupware* à equipe, já estão habituados a trabalhar neste sistema, e este deverá saber respeitar e se adaptar à organização adotada pelo grupo. Tammaro *et al.* [TAM 97] citam alguns papéis possíveis para o processo de escrita cooperativa: escritor, consultor, editor e revisor.

Já um modelo cognitivo consiste nas fases pelas quais o processo de trabalho em grupo se divide, englobando a descrição do comportamento dos participantes no decorrer do trabalho. Da mesma forma que a organização hierárquica, existem vários modelos cognitivos podem ser adotados pelo grupo, os quais vão representar a maneira como o grupo trabalha para atingir seu objetivo. Grande parte desses modelos apresentam, de alguma forma, fases para o planejamento, a coordenação e a negociação das atividades. No planejamento, são definidas as atividades que devem ser executadas para que o objetivo do grupo seja atingido e o papel de cada membro no decorrer do trabalho. A coordenação procura evitar que tarefas redundantes sejam executadas e permite que cada membro possa gerenciar suas atividades de acordo com os demais. E durante a negociação, busca-se resolver os conflitos gerados na realização das atividades, muitas vezes, através simplesmente da comunicação entre os membros [SAL 98], [DIE 96]. A Fig. 2.8 abaixo ilustra este ciclo de atividades (planejamento - coordenação – negociação).

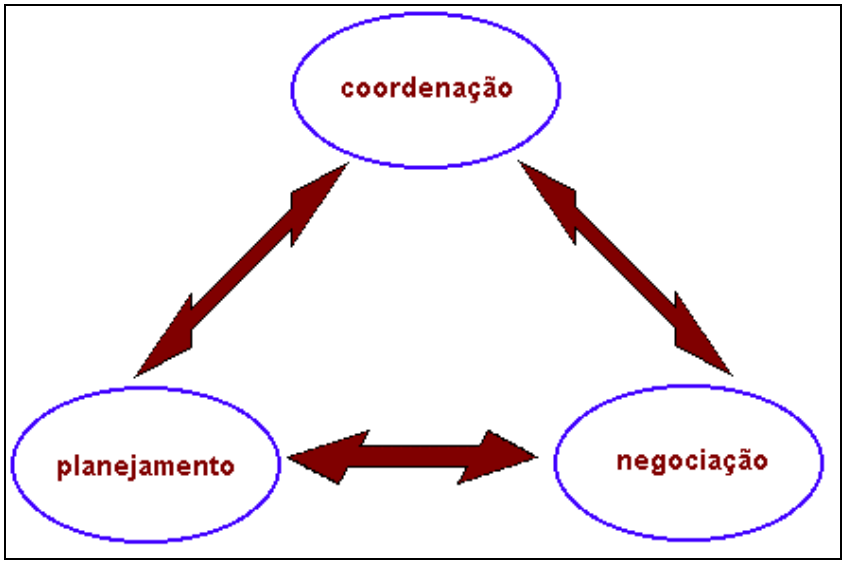


FIGURA 2.8 - Modelo cognitivo usual

Cada grupo utiliza, direta ou indiretamente, um modelo cognitivo, o qual conduz suas atividades até o objetivo final. Este modelo, que expressa a maneira como o grupo trabalha, bem como a organização hierárquica, que demonstra como o grupo se organiza, são aspectos sociais vitais para o sucesso de um *groupware*. Este deve obrigatoriamente respeitar estes aspectos, de acordo com o tipo de grupo e atividades pretende suportar e apoiar. Caso o *groupware* não respeite estes aspectos, correrá o risco de estar impondo uma nova organização ou um outro modelo ao grupo, causando mudanças talvez radicais na forma de trabalho deste, o que infringe diretamente alguns

dos requisitos básicos para sistemas de *groupware*, apresentados no início deste capítulo. Dessa forma, para poder atingir seus objetivos, sem infringir seus requisitos básicos ou recair em algum caso de insucesso, um *groupware* deve ser projetado para atender as necessidades de um tipo de grupo e atividade específicos, com uma organização hierárquica e modelo cognitivos bem definidos, para assim poder concentrar seus esforços naquilo que for realmente pertinente a este grupo. Um exemplo de ferramenta de *groupware* desenvolvida, focando-se um grupo específico de usuários, foi realizada pelo projeto POLITeam, onde todo o sistema foi projetado para uso interno do governo alemão [PRO 99], [PRI 99], [SOH 98].

2.2.2 Tecnologias envolvidas

Embora os aspectos sociais sejam decisivos para o sucesso de uma ferramenta de *groupware*, eles sozinhos não definem um groupware. Também, faz-se necessário analisar as tecnologias que vão viabilizar o suporte ao trabalho em grupo, adaptado às necessidades e características deste e do tipo de sistema que se deseja construir. Dentre estas tecnologias, pode-se destacar as redes de computadores e os sistemas distribuídos, os banco de dados e os sistemas operacionais, interfaces homem-máquina. agentes inteligentes e hipermídia. O modo como cada uma destas tecnologias é vista e utilizada dentro das pesquisas em CSCW é analisado a seguir.

- *Redes de Computadores e Sistemas Distribuídos:*

As redes de computadores são o meio sobre o qual são implantados os mecanismos de comunicação, compartilhamento e troca de informações [DIE 96]. Para Hollinsworth e Wharton [HOL 94], o próprio modelo cliente/servidor ganhou funcionalidade e popularidade com o crescimento de PCs e LANs como solução para organizar *workgroups* locais e de computação departamental. Entretanto, uma arquitetura para *groupware* precisa se expandir para além de sistemas locais. Segundo Siemienuch e Sinclair [SIE 94], as aplicações baseadas na premissa de altas velocidades e redes de comunicação públicas de baixo custo devem facilitar o trabalho cooperativo, ajudando a aumentar a qualidade e reduzir custos de projeto.

Uma arquitetura bastante comum em sistemas de *groupware* é a arquitetura cliente/servidor, a qual é uma forma eficiente e simples de estruturar um sistema como um grupo de processos cooperativos (servidores), que oferecem serviços para os usuários (clientes). Essa arquitetura permite otimizar o uso de recursos em clientes e servidor e inclui um certo grau de paralelismo, através da delegação de tarefas ao servidor, permitindo que o cliente possa se concentrar na interação com o usuário [DIE 96]. Entretanto, a centralização de um serviço sobre um único nó da rede pode ser uma forma de evitar duplicação de recursos, mas também constitui um ponto único de falha e um possível gargalo para o sistema, dependendo do tipo de serviço executado.

Para evitar esta centralização de serviços, a opção é o uso de conceitos de sistemas distribuídos. Como os usuários estão seguidamente distribuídos no tempo ou no espaço, muitos sistemas multiusuários são naturalmente considerados sistemas distribuídos, enfatizando a descentralização de dados e controles [ELL 91]. Um importante conceito de sistemas distribuídos é o conceito de comunicação de grupo, com o qual é possível a construção de servidores para facilitar tarefas como difusão de mensagens e a

notificação de eventos. Estas tarefas são essenciais para aumentar a confiabilidade do mecanismo de comunicação entre os membros do grupo, permitindo, por exemplo, que quando um evento ocorrer na área de trabalho de um usuário, ou quando este entrar em uma área de trabalho, todos os demais usuários ativos sejam notificados.

- *Banco de Dados e Sistemas Operacionais*

De modo geral, as tecnologias de banco de dados e sistemas operacionais são utilizadas para a gerência de dados em ferramentas de *groupware*. Alguns usos e requisitos para este processo de gerência podem ser considerados durante o processo de construção de um *groupware*:

a) Os usuários possuem objetivos individuais e do grupo;

b) Os usuários podem participar de múltiplas atividades no grupo;

c) Os usuários podem colaborar tanto síncrona quanto assincronamente, pois podem ser necessárias múltiplas sessões para que o objetivo seja atingido e talvez não seja possível agendar um horário comum a todos para a cooperação síncrona, ou pode um membro ter perdido uma sessão, sendo necessários mecanismos para que ele possa recuperar o trabalho perdido;

d) A colaboração pode envolver o compartilhamento de trabalho entre fronteiras de grupos;

e) Os usuários devem trabalhar eficientemente com o mínimo de distrações, sendo necessários mecanismos para filtrar as informações desnecessárias e apresentar resumos dos dados compartilhados e suas atualizações;

f) A colaboração sobre redes de longa distância, como a Internet, devem ser robustas contra falhas de clientes e devem fornecer o desempenho adequado mesmo quando os clientes têm conexões pobres.

Neste contexto, o uso de banco de dados oferece a CSCW suporte como repositório de informações, permitindo o acesso com facilidade a estas informações. Este suporte possibilita a manutenção da memória do grupo, a gerência de versões dos objetos compartilhados e o acesso concorrente a estes objetos [BOR 95], [BOR 95]. As soluções para o controle de concorrência devem procurar garantir tempos de resposta interativos e ainda fornecer consistência, o que pode envolver decisões de projeto entre tempo de resposta interativo, latência na propagação de atualizações, consistência ao apresentar os dados compartilhados e os tipos possíveis de atualização que os usuários podem fazer [PRA 99].

Entretanto, existem características dos sistemas de banco de dados que entram em conflito com características de sistemas *groupware*, tais como transações e o isolamento destas. O isolamento é usado em SGBDs para isolar as ações de um usuário dos demais e assim manter a consistência das informações, que são recuperadas de forma concorrente. Todavia, como já foi declarado anteriormente neste trabalho, a percepção dos demais membros do grupo e o compartilhamento de resultados parciais, são pontos importantes para sistemas de *groupware*.

Quanto às transações, o ambiente de um *groupware* caracteriza-se por transações de longa duração, que dificultam o processo de *roolback*, e por um controle interativo, o que torna inconveniente um escalonamento prévio das transações e seu *redo*. Para a solução destes problemas, a abordagem mais aceita tem sido o modelo de *check-*

in/check-out, onde o usuário faz uma cópia do objeto para sua área particular (*check-out*) e lá executa suas alterações. Concluídas estas alterações, a nova versão é depositada novamente no banco de dados (*check-in*). Sob esta visão, cada objeto passa, então, a ser uma coleção de versões, com mecanismos de reserva e cópias de versões. Outro mecanismo passível de ser utilizado é o mecanismo de notificação. Neste, o usuário é incluído como parte do controle de concorrência da seguinte forma: sempre que um participante do grupo tenta acessar um objeto que está sendo utilizado por outro participante, ambos são notificados e podem, então, interagir diretamente para resolver o confronto [BOR 95], [DIE 96].

Já os sistemas operacionais oferecem funcionalidades básicas voltadas ao trabalho cooperativo, tais como o controle a recursos compartilhados entre participantes locais e remotos e mecanismos de comunicação. Entre as vantagens de se ter estas funcionalidades, pode-se citar o aproveitamento de funções já existentes neste tipo de sistema, relacionadas principalmente ao acesso concorrente e à gerência de recursos distribuídos, determinando um acesso consistente e eficaz aos recursos, levando em conta diversos fatores como segurança, transparência de localização e tolerância a falhas [DIE 96].

- *Comunicação*

A tecnologia de comunicação em CSCW enfatiza a troca de informações entre agentes remotos e um de seus desafios é "como tornar interações distribuídas tão efetivas quanto interações face a face" [ELL 91]. Uma interação remota, suportada por tecnologias adequadas, apresenta um meio alternativo, mas não irá substituir a interação face a face, embora possa ser preferível em certas situações, devido a certos inconvenientes que podem ser minimizados. "O desafio, então, é aplicar as combinações tecnológicas apropriadas para as classes de interações que serão mais beneficiadas pelo novo meio" [ELL 91].

Esta comunicação também pode se dar através do compartilhamento de informações. Quanto maior e efetivo compartilhamento de informações entre os membros do grupo, maior será o resultado da cooperação. Este efetivo compartilhamento depende de mecanismos de coordenação para controlar e propagar as alterações. Para facilitar este compartilhamento pode-se replicar, em parte ou por completo, os depósitos de informações por uma comunidade de servidores, fornecendo a cada usuário acesso a uma única imagem da informação mantida no servidor local, cuja consistência pode ser mantida pela sincronização periódica de servidores. Entretanto, isto levanta novamente os pontos discutidos quanto ao uso da tecnologia de banco de dados, utilizados para armazenar as imagens nos servidores locais, e ainda, os problemas relacionados a redes de computadores, que interconectariam os servidores, e poderiam sofrer um *overhead* considerável com as sincronizações, dependendo do tamanho da rede e da dinamicidade dos objetos [HOL 94].

- *Interface Homem-Máquina (HCI _ Human-Computer Interface)*

Os estudos na área de Interface Homem-Máquina é (assim como CSCW) um campo multifacetado e sua aplicação em ferramentas de *groupware* enfatiza a importância da interface com o usuário nestes sistemas [ELL 91]. Para Dietrich [DIE 96], a interface do sistema deve preferencialmente encorajar a cooperação, dando ciência das atividades dos demais participantes e podendo se adaptar aos requisitos do

grupo. Isto porque um dos objetivos que deve ser perseguido no desenvolvimento deste tipo de sistema é a praticidade da interface. A interface de um *groupware* deve, idealmente, ser útil, enfatizando a interação e o compartilhamento das atividades, pois esta deve ser vista principalmente como um caminho de acesso aos demais membros do grupo.

- *Inteligência Artificial e Agentes Inteligentes*

Um *groupware* projetado para uso de diferentes grupos precisa ser flexível e acomodar uma variedade de procedimentos e tarefas. A inteligência artificial pode, a longo prazo, fornecer significantes contribuições, transformando a máquina, de agente passiva, que processa e apresenta as informações, para agente ativo, que aprimora as interações [ELL 91]. Os agentes inteligentes, que podem ser vistos como "entidades autônomas para solução de problemas agindo em cooperação e coordenação para alcançar seus objetivos" [JEN 98], podem auxiliar neste processo. O seu uso concentra-se principalmente na utilização de agentes para a realização de tarefas que seriam tediosas ao elemento humano, mas que são importantes para a atividade cooperativa, como a criação de boletins informativos e a notificação da presença de participantes [DIE 96]. A aplicação efetiva de agentes inteligentes em sistemas intensificou-se na década de 90 e um exemplo destes agentes em um ambiente cooperativo é encontrado em [SCH 98], onde os participantes são representados no sistema por um conjunto de agentes, que executam (ou antecipam) para o usuário várias tarefas.

- *Hipertexto e Multimídia*

As tecnologias de Multimídia e Hipertextos são de grande interesse para a área de CSCW. O uso destas tecnologias, intimamente ligadas à tecnologia de interface homem-máquina, discutida anteriormente, permite a criação de interfaces mais convenientes e funcionais para ferramentas de *groupware*, especialmente quando um grande número de informações a ser compartilhadas está disponível.

O uso da multimídia, por exemplo, fornece um meio de interação mais natural entre os participantes, através de recursos como músicas, imagens e vídeo, que são elementos de comunicação universal. O problema no uso destes objetos multimídia encontra-se principalmente na digitalização destas mídias. Este processo pode ocupar grande espaço em disco e memória, mesmo com o uso de algoritmos de compressão, o que dificulta bastante o seu transporte por redes de comunicação de baixo desempenho. Já o enfoque de hipertextos em CSCW permite a criação e interligação de fragmentos de informação [DIE 96].

Um hipertexto ou hiperdocumento consiste num texto ou documento não-linear, formado por um conjunto de nós e uma rede de ligações entre eles. Cada nó pode ser um trecho de informação do documento que representa um conceito ou idéia, enquanto que as ligações conectando estes nós representam os relacionamentos, denotando a associação de informações entre estes nós [SOU 96] [DIE 96].

Dentre as vantagens no uso de hiperdocumentos, pode-se citar a facilidade de navegação entre referências, a possibilidade de estruturar as informações e de criar novas referências e anotações e a modularização das informações, imposta pelo uso de nós [DIE 96]. Estas vantagens fazem com que a união entre CSCW e hiperdocumentos traga benefícios tanto a sistemas hipermídia, resultando em sistemas hipermídia multi-

usuários, quanto às atividades de cooperação e colaboração, fornecendo suporte a documentos estruturados [PIM 98]. O enfoque de hipertextos auxilia na criação de fragmentos interligados, através de nós, o que permite ao sistema estabelecer associações entre as informações geradas por diferentes membros do grupo, enquanto que as várias formas de mídia vão, por sua vez, estender a capacidade de expressão do grupo [BOR 95]. O próprio conceito e espantoso desenvolvimento da WWW _ Wold Wide Web _ acentuou ainda mais o interesse por essa união, destacando-se a criação de algumas ferramentas de *groupware* como fruto deste interesse, como o BSCW [GMD 00b],[GMD 00a] e o Byzance [OPE 99].

2.3 Considerações Finais

Neste capítulo viu-se os conceitos e as tecnologias relacionados à área de CSCW, incluindo classificações e requisitos para ferramentas de *groupware*. Todos estes conceitos são vitais para o desenvolvimento e para a compreensão destas ferramentas, sendo também importantes para uma clara e completa compreensão deste trabalho. Com base nestes conceitos, o próximo capítulo segue os estudos dentro da área de CSCW, capturando e analisando a fundo o domínio da percepção (*awareness*), tema central sobre o qual este trabalho se desenvolve.

3 *Awareness* em Sistemas de *Groupware*

Como já foi comentado no capítulo anterior, a procura por uma maior eficiência na solução de problemas cada vez mais complexos tem feito com que atividades, antes individuais, passassem a ser resolvidas agora em grupos de trabalho, com cada membro contribuindo com sua parte, para chegarem ao produto final [SPU 94]. Entretanto, a idéia de cooperar, de "trabalhar ou agir em conjunto para um objetivo comum" não é uma tarefa simples [BEL 95]. Um problema muito comum que pode prejudicar o trabalho em grupo é a falta de contexto entre os participantes. Esta ocorre quando os membros de um grupo desconhecem o que seus colegas estão fazendo, ou não sabem onde suas próprias atividades se encaixam no trabalho como um todo, nem qual é a situação atual desse trabalho. Esta falta de contexto sobre as atividades dos colegas e do grupo pode gerar uma série de inconvenientes que reduzem a eficiência e a qualidade do trabalho em grupo, como redundâncias nas tarefas, inconsistências e contradições.

Este capítulo trata especificamente desta questão: a falta de contexto sobre as atividades do grupo, a qual só pode ser evitada graças ao conceito de percepção ou, do original em inglês, *awareness*. A percepção envolve saber quem é o grupo, qual seu objetivo e suas atividades. Envolve o conhecimento sobre o que aconteceu, o que vem acontecendo e o que está se passando agora dentro das atividades do grupo; sobre quem são os membros deste grupo, onde estão e o que estão fazendo.

O conceito de percepção é chave para a definição e compreensão desta dissertação. Sendo assim, este capítulo se destina a esclarecer este conceito, dividindo-se em 3 seções para melhor apresentá-lo. Na primeira seção, será dado o conceito de percepção propriamente dito, enfatizando-se sua necessidade em ferramentas de *groupware*. Na segunda seção, propõe-se um esquema de classificação para o domínio da percepção, sobre o qual são dispostas, na terceira seção, diversas ferramentas de *groupware* hoje disponíveis na bibliografia. Esta classificação se faz necessária, pois o domínio da percepção envolve uma gama muito grande de informações para o desenvolvimento de um suporte adequado. Outras vantagens desta classificação são auxiliar na melhor identificação do problema tratado nesta dissertação, que será apresentado em detalhes no próximo capítulo, e apresentar um resumo das contribuições mais significativas encontradas na bibliografia até o momento, o que, por si só, representa uma importante contribuição ao domínio da percepção,.

Convém aqui mencionar que este capítulo foi elaborado a partir de diversos artigos escritos anteriormente, como Pinheiro et. al. [PIN 00a],[PIN 00b] e Pinheiro e Lima [PIN 99b]. Os conhecedores destes artigos encontrarão grande semelhança entre o texto deste capítulo e o apresentado nestes artigos. Dessa forma, este capítulo, bem como os artigos, constituem uma importante contribuição desta dissertação à comunidade de CSCW, devido à classificação geral sobre o tema percepção apresentada, a qual resume o conteúdo e as direções das mais significativas contribuições neste domínio.

3.1 Percepção

A falta de contexto dentro de um grupo pode causar uma série de problemas capazes de afetar a desejada eficiência e qualidade do trabalho em grupo. Quando os membros não têm conhecimento sobre o que está sendo desenvolvido pelos seus colegas, o trabalho resultante pode ser truncado, sem coesão, não representando as idéias do grupo como um todo, mas somente um conjunto de idéias soltas, com pouca ou nenhuma ligação entre elas, ou ainda incluir inconsistências ou até mesmo contradições. A presença de inconsistências e a falta de coesão reduzem sensivelmente a qualidade do trabalho, fazendo com que seu resultado seja formada apenas por contribuições isoladas e não por um conjunto consistente de qualidade superior, afetando diretamente os objetivos do trabalho em grupo. Um exemplo disso pode ser encontrado durante a edição cooperativa de um documento. Esta é uma atividade intrinsecamente cooperativa, uma vez que um grupo de autores poderá produzir melhores resultados, e de modo mais eficiente, se combinarem suas habilidades, pelo confronto de suas experiências, de seu nível de especialização e pelas contribuições de cada membro [SAL 98]. Entretanto, quando um autor desconhece o que seus colegas estão escrevendo, ele poderá incluir idéias que já foram tratadas por outros autores, idéias que em nada contribuem ou estão pouco relacionadas ao texto e até mesmo incluir posições contrárias às apresentadas por seus colegas no restante do documento, podendo gerar um documento impreciso, contraditório ou sem fundamento. Um documento assim acaba consumindo muito tempo de revisão, baixando a eficiência do processo de edição, ou pode ainda causar conflitos entre os autores para a resolução das posições desconexas incluídas no texto. Como conseqüência, o conhecimento do estado ou das ações dos colaboradores se torna requisito para a escrita cooperativa [BAE 93].

Torna-se claro, então, que estar atento aos colegas e às atividades por eles desempenhadas representa papel importante na fluidez e na naturalidade do trabalho [GUT 99]. Para tanto é necessária a definição de um contexto para o grupo e suas atividades. Este contexto não se limita somente ao conteúdo das contribuições individuais, mas também atinge o seu significado para o grupo como um todo, bem como seu objetivo [BOR 95]. Ao fornecimento deste contexto aos membros de um grupo se dá o nome de percepção (*awareness*). A percepção pode ser conceituada como sendo a contextualização das atividades individuais através da compreensão das atividades realizadas por outras pessoas [ARA 97], ou ainda, como o conhecimento geral criado pela interação entre um agente e seu ambiente, ou simplesmente, saber o que está acontecendo, envolvendo o estado do conhecimento e os processos de perceber e agir [GUT 99]. Em outras palavras, percepção refere-se a ter conhecimento, ter ciência das atividades do grupo, das atividades que influenciarão o trabalho como um todo. Envolve saber o que tem acontecido, o que está acontecendo e/ou o que poderá vir a acontecer dentro do trabalho e do ambiente de trabalho de um grupo, além do próprio conhecimento do que é este trabalho (quais são as atividades que devem ser realizadas para que este seja concluído, quais são os prazos, etc.) e do grupo (quem são seus membros, quais suas funções, o que estão fazendo, etc.). É através das informações oriundas da percepção que é possível responder questões como “quem realizou dada tarefa? quando?”, “quem está trabalhando agora? em que está trabalhando?”, “quem é o responsável por uma tarefa?” e “o que ainda falta ser feito?”. Generalizando, pode-se afirmar que "*awareness* significa uma compreensão do estado total do sistema, incluindo atividades passadas, *status* atual e opções futuras" [SOH 98].

A Fig. 3.1 abaixo ilustra os problemas decorrentes da falta de mecanismos de percepção adequados. Na figura, observa-se um grupo que não interage entre si, onde os membros não têm idéia do que está sendo realizado pelos colegas, causando uma série de problemas. Por exemplo, os dois membros colocados à esquerda do desenho estão depositando contribuições contrárias (os itens "-A" e "+A"), um sem conhecimento da contribuição do outro. A direita, dois membros estão sendo redundantes, depositando contribuições idênticas ("B"). Tem-se ainda um membro perdido, com muitas dúvidas sobre o trabalho que o grupo vem realizando, como o membro no canto inferior direito, que deposita uma questão sem obter resposta, e ainda um membro depositando contribuições desconexas (como a contribuição "7"), por problemas de equivocalidade [DAF 86]. Estas situações podem afetar negativamente o trabalho e poderiam ser amenizadas com mecanismos adequados de suporte à percepção. Uma ferramenta de *groupware* que não apresenta um suporte adequado à percepção não está se prevenindo quanto a estas situações.

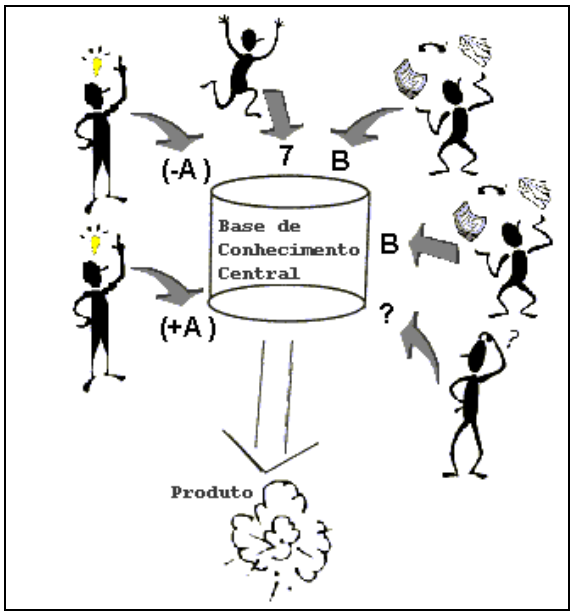


FIGURA 3.1 - Um grupo de trabalho sem suporte a *awareness*

A percepção é peça chave para qualquer forma de cooperação, uma vez que perceber, reconhecer e compreender as atividades dos outros é um requisito básico para a interação humana e a comunicação em geral [SOH 98]. Por exemplo, a interação em um mesmo local tem alto grau de percepção dos colegas. Pode-se facilmente saber se um colega está no escritório, se está trabalhando em um projeto ou se está de mau humor e, com o tempo, aprende-se também sobre sua agenda, hábitos e interesses. Tudo isso contribui para conhecer melhor os colegas, criando uma base para as interações, sem a qual estas ficam mais formais e menos fluídas [HUD 96]. A percepção permite a cada usuário coordenar e estruturar seu trabalho, pois possibilita que o mesmo possa perceber e compreender no que os demais estão trabalhando. Ela também mostra oportunidades para comunicação informal e espontânea e suporta o estabelecimento e a manutenção de convenções no grupo. Sem a percepção, o trabalho cooperativo coordenado é quase impossível [SOH 98]. Considerando, por exemplo, o caso do desenvolvimento cooperativo de *software*, é necessário que os membros tenham noção

do contexto de suas atividades no contexto geral do processo para que possam perceber o andamento das atividades realizadas pelos demais e compreender como os resultados gerados por estas atividades podem ser conjugados com os seus próprios, para mais rapidamente chegarem ao resultado final [ARA 97]. Sem um suporte adequado à percepção, neste caso, fica quase impossível produzir um resultado (*software*) consistente e de qualidade, de forma eficiente. Em resumo, a percepção é muito importante para melhorar a eficiência e a compreensão do processo de cooperação, evitando uma edição competitiva [DIA 97].

A informação de percepção é, segundo Mariani [MAR 97], fornecida por um conjunto de mecanismos os quais dão ao usuário alguma indicação das atividades dos demais. Estes mecanismos fornecem oportunidade aos indivíduos de ganharem uma compreensão do trabalho de seus colegas e usar esta informação para coordenar suas atividades para o grupo [DOU 97]. Como se pode observar pelo comentado nos parágrafos anteriores, o conceito de percepção envolve um vasto conjunto de informações, como atividades, prazos e informações sobre os colegas. Isto faz com que, usualmente, os mecanismos de suporte apresentados na literatura tenham abordado somente um subconjunto destas informações, implementados através de uma série de abordagens, como recursos de interface e notificações. Com isso, tem-se identificado dentro da bibliografia um conjunto limitado de características importantes, que vem sendo observado nas propostas destes mecanismos. A identificação deste conjunto permitiu a delimitação de um esquema de classificação com estas características, consideradas chaves para o suporte à percepção e que idealmente deveriam ser observadas pelas ferramentas de *groupware*. Sobre este esquema de classificação apresentado na próxima seção, pode-se distribuir os mecanismos de suporte à percepção hoje mais difundidos, criando assim uma espécie de classificação geral para o assunto, a qual poderá também ajudar na condução de novas contribuições. A classificação aqui proposta age também como uma reunião das principais contribuições hoje encontradas na bibliografia, fornecendo uma visão geral do que vem sendo desenvolvido nos últimos anos no domínio da percepção. A seguir, então, encontra-se a discussão sobre as características identificadas na literatura, juntamente com o esquema de classificação proposto.

3.2 Uma classificação para percepção

Como já foi discutido anteriormente, devido a sua destacada importância, vários mecanismos de suporte à percepção vêm sendo propostos na bibliografia. A análise destes mecanismos mostrou uma série de características importantes para este suporte. Estas características giram em torno de seis questões: *o que, quando, onde, como, quem e quanto*. Cada uma destas questões identifica aspectos vitais para o fornecimento de percepção dentro de um *groupware*. O próprio *groupware* influencia fortemente nestas questões, uma vez que as necessidades em *groupware* síncronos são diferentes das necessidades em *groupware* assíncronos. Não se trata somente da questão do tempo, de estar ou não trabalhando simultaneamente, mas do que se precisa em um ambiente de trabalho síncrono e em um ambiente de trabalho assíncrono. Em um ambiente síncrono, há a necessidade dos membros estarem todos trabalhando simultaneamente, de estarem todos conectados ao sistema em um mesmo momento (a interação síncrona descreve a situação onde mais de um usuário está acessando de modo concorrente a dados

compartilhados [MAR 97]). Exemplos típicos são sistemas de áudio e videoconferências. Esta necessidade já não ocorre em sistemas assíncronos, onde pode haver um intervalo de tempo entre a atuação de um usuário e sua percepção por seus colegas (a interação assíncrona ocorre quando os usuários estão compartilhando um objeto sobre um período estendido [MAR 97]). Estes são sistemas onde os usuários não precisam estar todos trabalhando simultaneamente para que o objetivo seja atingido. Neste caso, o fato de dois ou mais membros estarem conectados ao mesmo tempo é encarado apenas como coincidência, uma oportunidade a mais de interação e cooperação, mas não uma obrigatoriedade.

Com isso, os *groupware* síncronos e assíncronos diferem quanto as suas necessidades por percepção, uma vez que usuários obrigatoriamente trabalhando ao mesmo tempo terão necessidades de percepção diferentes daqueles que não precisam trabalhar simultaneamente. Dessa forma, torna-se importante, ao analisar as seis questões mencionadas acima (o que, quando, onde, como, quem e quanto), observar qual a situação empregada, se é um ambiente síncrono ou assíncrono.

3.2.1 O que

A primeira questão a ser analisada é "o que" e se refere a quais as informações devem ser fornecidas aos usuários. Estas informações estão intimamente ligadas a dois aspectos principais: as atividades e os papéis. As atividades estão explicitamente colocadas dentro da própria definição de percepção (ter conhecimento das atividades do grupo), enquanto os papéis desempenhados pelos membros do grupo vão influenciar em quais informações serão de interesse desses mesmos membros.

As atividades são a base do trabalho cooperativo. O objetivo a ser alcançado pelo grupo, na maioria dos casos, é dividido em atividades menores, repartidas entre os membros do grupo. Esta divisão de responsabilidades, seja ela implícita ou explícita, deve ser de conhecimento do grupo, para que os membros possam saber o que seus colegas estão fazendo. Esse saber é um conhecimento importante, tanto em sistemas síncronos quanto em assíncronos, mas, por vezes, de maneiras distintas. Em um ambiente síncrono, é mais interessante saber em detalhes quais as atividades que estão sendo realizadas no momento, enquanto que em um sistema assíncrono, como não há garantias quanto a em que momento determinada tarefa será realizada por um colega, é mais interessante saber quem está responsável por quais atividades, que trecho do trabalho está sob os cuidados de quem, etc. Por exemplo, dificilmente seria interessante para alguém em um ambiente assíncrono saber que um objeto no espaço de trabalho compartilhado, como uma classe em um diagrama de classes, foi movido de seu lugar original, especialmente porque, até que a informação seja percebida pelos demais membros, dias podem ter se passado. Isto já não ocorre em um sistema síncrono, onde esta mesma informação está inserida dentro do contexto das atividades que estão sendo realizadas no momento, sendo importante a percepção deste contexto por todos os participantes ativos.

Assim, pode-se dizer que, em ambientes síncronos, o maior interesse está em informações detalhadas sobre as atividades, nestas vistas em um "micro-nível", como em um microscópio. Isto porque os membros estão trabalhando ao mesmo tempo em um mesmo ambiente de trabalho, então, torna-se importante que eles percebam as

atividades de seus colegas nos mínimos detalhes. Por exemplo, em uma reunião é importante que os participantes estejam atentos a qualquer contribuição dos colegas, seja ela uma nova questão, hipótese ou anotação em um *whiteboard*. Outro exemplo são as ferramentas de *groupware* para diagramação. Nestes, é importante que os participantes sejam capazes de perceber pequenas alterações no ambiente, como a remoção de parte do diagrama, introdução de novos elementos, ou alterações, mesmo que de posição, nos já existentes, para que o contexto das atividades seja mantido.

O mesmo já não ocorre em ambientes assíncronos, onde um nível de detalhamento tão grande não é tão adequado, podendo mais atrapalhar o trabalho dos membros do que ajudar. Nestes ambientes, devido à existência de um possível espaço de tempo entre as atividades dos membros, é mais importante ter uma idéia global das atividades realizadas pelos colegas do que saber em detalhes, como se uma dada classe em um diagrama de classes foi movida de seu lugar anterior, ou se um ponto foi substituído por uma vírgula em um texto. Em outras palavras, é mais importante fornecer uma visão em "macro-nível", mais global, das atividades quando em um ambiente assíncrono. Neste caso é mais interessante saber, por exemplo, quem está responsável por uma atividade, desde quando e qual o prazo limite; ou sob responsabilidade de quem está concluir um certo capítulo do texto.

De modo geral, em ambientes síncronos, informações como posição dos cursores, movimentos do *mouse* e a própria posição dos colegas no espaço de trabalho compartilhado são informações interessantes para se compreender o contexto das atividades que estão sendo realizadas no momento. Já em ambientes assíncronos, um detalhamento maior das atividades não se faz necessário, mas, em compensação, fornecer um resumo significativo é vital para a percepção destas pelo grupo. Neste último caso, é mais interessante ter acesso a informações como o *status* das atividades (quanto já está concluído), um resumo das últimas contribuições significativas ou a descrição das atividades.

Quanto aos papéis, estes são elementos de destaque dentro de um ambiente cooperativo, pois representam a existência de uma hierarquia, de uma estrutura de autoridade, dentro do grupo. Eles estão intimamente ligados ao tipo de atividade, responsabilidade e direitos que poderão ser efetuadas ou assumidas por cada membro do grupo. Por exemplo, um membro que somente pode desempenhar o papel de "leitor" durante a edição cooperativa de um texto não poderá fazer nada além de sugestões e críticas ao texto, enquanto que um membro que possa também desempenhar o papel de "escritor", além de dar sugestões, poderá modificar o texto e discutir estas mudanças com os demais colegas.

Devido a esta ligação entre os papéis e as possíveis atividades dos membros, estes acabam por influenciar também no suporte à percepção, uma vez que os papéis desempenhados pelos membros vão fornecer uma boa indicação sobre o tipo de informação de percepção que será necessária para que este papel possa ser desempenhado com sucesso. Além disso, os papéis também podem indicar quais as informações de percepção podem ou não ser fornecidas para cada membro. A maior diferença encontra-se nas informações necessárias para um participante qualquer e para aqueles que desempenham função de chefia ou liderança, o chamado "coordenador". A coordenação é característica fundamental em atividades cooperativas e o fornecimento de informação de percepção para a coordenação é importante, tanto em sistemas síncronos quanto assíncronos [DOU 97].

Sem um suporte à percepção específico para a coordenação, esta atividade se torna ainda mais complexa, dificultando o trabalho do coordenador e podendo vir a prejudicar a eficiência da colaboração [RAP 99]. Um coordenador precisa de informações de percepção mais completas que as de um simples participante, pois ele precisa ter percepção do grupo como um todo. O coordenador precisa estar atento ao andamento do trabalho como um todo, desde as atividades de cada membro (quais foram concluídas, quando, quantas ainda faltam, qual o estado atual de cada uma), até o somatório de todas elas, para ter uma visão global do grupo. Ele, mais do que ninguém, precisa de informações como prazos, datas e *status* das atividades, a fim de poder dirigir os esforços da equipe para os pontos onde estes são mais necessários. É com base neste tipo de informação que o coordenador poderá tomar decisões importantes como, por exemplo, encerrar a discussão e partir para a votação sobre um tópico em uma reunião, fazendo com que seja vital que estas informações sejam fornecidas, tanto em sistemas síncronos quanto em assíncronos. Se o coordenador do grupo tem disponíveis mecanismos de percepção próprios, com os eventos relevantes ao seu papel, achará mais fácil a tarefa de conduzir o grupo a um resultado bem sucedido [BOR 99b].

Um exemplo deste tipo de percepção diferenciada de acordo com o papel é apresentado na Fig. 3.2, onde o coordenador tem uma visão geral da participação dos membros numa discussão através desta implementação conhecida como “Participameter”. O Participameter mostra o nível de participação dos membros como uma percentagem da participação total [BOR 99b]. Na Fig. 3.2 é mostrada uma implementação do Participameter, onde é possível perceber detalhes das percentagens de cada elemento (“contrib”, “reads” e “remarks”) para cada contribuição, na parte inferior da figura. Observa-se também a associação entre valores percentuais e um padrão de cores, permitindo uma rápida assimilação da informação.

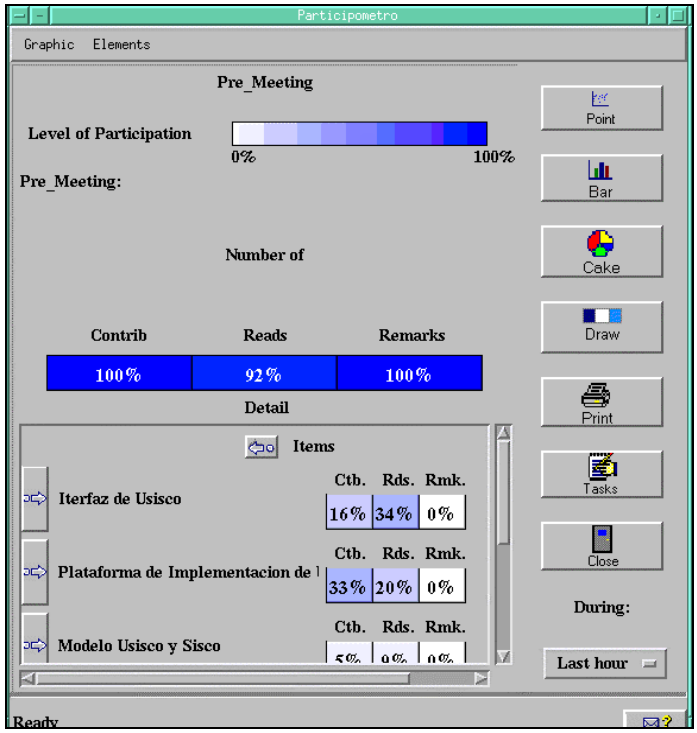


FIGURA 3.2 - Implementação do Participameter [BOR 99b]

Todavia, convém mencionar que, embora sejam vitais para a coordenação, informações como o *status* do trabalho total ou de cada atividade também podem servir aos participantes, especialmente em ambientes assíncronos, para dar a noção de grupo ao participante, tirando-lhe a impressão de estar trabalhando sozinho e contribuindo para movê-lo da inércia. A sensação de solidão e a inércia são problemas comuns em ambientes assíncronos, quando não há um suporte adequado à percepção. Nestas situações, ocorre que os membros, por não estarem atentos às atividades dos demais, têm a sensação de estarem sozinhos, ou de que o trabalho não está progredindo, permanecendo, muitas vezes, em um estado de inércia, agravado também pelo desconhecimento da importância de suas atividades para o grupo, de seus prazos ou do andamento dos trabalhos.

Resumindo, então, ao fazer a pergunta "o que", identificam-se duas características importantes no suporte à percepção: as atividades e os papéis, os quais trazem que informações devem ser apresentadas para o suporte adequado à percepção. A Tab. 3.1 abaixo apresenta o resumo destas características que devem ser observadas quando se faz a pergunta "o que".

TABELA 3.1 - Características para a pergunta "o que"

		Ambientes Síncronos	***Ambientes Assíncronos***
O que	*Atividades*	Micro-nível	Macro-nível
	Papéis	Diferenciação das informações de acordo com o papel desempenhado	

3.2.2 Quando

A segunda questão que deve ser feita ao se pensar em percepção é "quando". Esta questão refere-se a dois aspectos essenciais: quando ocorreram os eventos geradores das informações de percepção e quando se dá a apresentação destas informações aos usuários (quando fornecer a percepção).

Assumindo que as informações de percepção são geradas por eventos que ocorrem durante o trabalho do grupo, o momento de ocorrência destes torna-se característica importante para a percepção, pois de acordo com o momento, estes eventos vão ser mais ou menos úteis à percepção em ambientes síncronos e assíncronos.

De modo geral, pode-se dividir a ocorrência dos eventos em quatro momentos-chave: o passado, para eventos que ocorreram em um intervalo de tempo no passado; o passado contínuo, para eventos que começaram no passado, mas que continuam válidos até agora; o presente, para eventos que estão ocorrendo neste momento e o futuro, representando as opções futuras para o grupo. O passado inclui aqueles eventos que já ocorreram há algum tempo e cujos resultados podem não ser mais válidos. Por exemplo, uma alteração feita em um texto que já foi substituída por outra modificação. O passado contínuo representa aqueles eventos que, apesar de terem iniciado no passado, continuam valendo até o presente. Por exemplo, um membro do grupo que ficou responsável por uma tarefa, a qual ele ainda não concluiu, tampouco outro membro a

tenha assumido. Já o presente é composto estritamente pelos eventos que estão acontecendo agora, tais como o movimento do *mouse*, uma anotação em um *whiteboard* ou a alteração da posição de uma barra de rolagem. Por fim, o futuro são as opções do grupo, os eventos que poderão ocorrer, mas que precisam fazer parte da percepção do usuário. É o caso, por exemplo, de um alarme que avise quanto à aproximação dos prazos para as atividades. A chegada do prazo limite, que é o evento gerador, só ocorrerá no futuro, mas o alarme que fornece a percepção deste evento se dá antes.

Alguns destes momentos são mais ou menos importantes para sistemas síncronos ou assíncronos. Em ambientes síncronos, por estarem os membros trabalhando ao mesmo tempo, é vital fornecer a percepção dos eventos que estão ocorrendo agora, no momento presente. Sem esta percepção do presente, os participantes podem ficar totalmente perdidos, sem a menor noção do que está havendo dentro do trabalho em grupo. Já em ambientes assíncronos, como há um possível espaço de tempo entre a atuação dos colegas, é vital manter o contexto sobre o que foi feito (eventos no passado) e do que ainda está sendo feito (passado contínuo), para que os participantes possam encaixar suas próprias atividades no contexto das atividades do grupo. Por exemplo, durante o trabalho assíncrono, um membro, ao ingressar no sistema, precisa saber o que foi feito em sua ausência antes de dar continuidade a seu trabalho. Entretanto, em ambos os casos, tanto sistemas síncronos quanto assíncronos, a percepção de eventos futuros representa uma opção interessante para manter os membros atentos aos possíveis rumos que o trabalho poderá tomar. Por exemplo, em ambientes assíncronos, manter a percepção dos prazos das atividades contribui para minimizar a inércia nos participantes ao mantê-los conscientes da aproximação das datas limites.

Obviamente, não se pode excluir a possibilidade de fornecer a percepção quanto a eventos no passado e no passado contínuo (especialmente este último) em ambientes síncronos e da percepção para o momento presente em assíncronos. Todavia, estas representariam apenas opções a mais, novas oportunidades para a cooperação nestes sistemas e não uma necessidade, sem as quais os mesmos sairiam prejudicados.

O interesse pelos eventos ocorridos em um ou outro momento vai determinar outra importante característica para suporte à percepção, a persistência ou a utilidade: são as informações de percepção persistentes? Até quando elas são úteis para o trabalho do grupo? Em sistemas síncronos, como o interesse maior se concentra no momento atual, não há a necessidade de manter as informações de percepção armazenadas por muito mais tempo do que aquele momento, pois passado este, elas não serão mais úteis ao grupo. Logo, há a necessidade de uma baixa persistência. Já em sistemas assíncronos, dá-se o contrário. Nestes sistemas, o grande interesse reside na percepção de eventos no passado ou no passado contínuo, logo, há a necessidade de um armazenamento das informações colhidas nestes eventos no mínimo até o momento em que os usuários se tornarem cientes destas. Em outras palavras, as informações geradas continuam sendo úteis ao grupo, mesmo tendo passado o momento de sua ocorrência, havendo, então, a necessidade por uma alta persistência destas informações.

Entretanto, deve-se determinar um limite de tempo para este armazenamento em ambientes assíncronos. Deve-se determinar até quando estes eventos serão úteis para a percepção dos usuários, antes de se tornarem apenas um incômodo às atividades dos mesmos. Se só interessarem aqueles eventos que ainda são válidos (passado contínuo), estes deixam de ser úteis e podem ser descartados assim que deixarem de ser verdadeiros, por exemplo, assim que as atividades forem sendo concluídas. Mas se há o interesse por eventos passados, estes devem ser persistentes até um certo ponto e um critério para definir a caducidade destas informações deve ser determinado. Este critério

pode ser, por exemplo, um tempo de vida fixo (*time life*), ou se todos os membros (ou um membro em especial) já foram informados sobre o ocorrido (se a informação já foi fornecida), ou qualquer outra condição aplicada aos eventos geradores.

Outra característica importante com relação à questão "quando" é quando fazer a apresentação das informações. Em outras palavras, deve-se decidir quando dar ao usuário a informação de percepção. Em sistemas síncronos, como o interesse está nos eventos que estão ocorrendo agora, quanto mais cedo for dada a percepção destes eventos ao grupo, melhor (imediato), o que também significa que as informações retiradas destes eventos não precisam ser persistentes, conforme discutido acima. Já em sistemas assíncronos, por definição, há um intervalo de tempo entre os eventos geradores e os demais membros ingressarem no sistema e perceberem o que foi realizado. Por definição, então, há um intervalo entre a ocorrência do evento e a sua percepção pelos colegas do grupo. Assim, nestes sistemas, a informação da percepção, além de ser persistente (como discutido anteriormente), é apresentada aos usuários possivelmente em um momento posterior à ocorrência dos eventos, já que não há garantias quanto a este intervalo de tempo entre a ocorrência e a percepção pelos usuários. Nestes sistemas, o mais adequado é permitir ao próprio usuário do sistema decidir em que momento desejará receber as informações de percepção sobre o que aconteceu e vem acontecendo no grupo.

A Tab. 3.2 abaixo resume o que foi discutido aqui sobre a questão "quando": quando ocorrem os eventos, até quando as informações destes são úteis e quando apresentar estas informações, com as observações sobre os critérios de apresentação e caducidade colocadas entre parênteses.

TABELA 3.2. - A questão "quando" para percepção

		Ambientes Síncronos	***Ambientes Assíncronos***
Quando	*Eventos*	Presente	Passado Passado Contínuo
		Futuro	Futuro
	Apresentação	Imediata	Posterior (a critério do usuário)
	Persistência/Tempo de Utilidade	Baixa	Alta (critério de caducidade)

3.2.3 Onde

A terceira questão, que deve ser feita quando o assunto é percepção, é "onde" e se refere a onde as informações são geradas e apresentadas. Essa é uma questão que envolve essencialmente o espaço de trabalho e os objetos compartilhados. Ao se trabalhar em um grupo, assume-se que se trabalha sobre um conjunto de objetos dispostos dentro de um espaço de trabalho, compartilhados entre todos os membros do grupo, que vão aí desenvolver suas atividades e interagir entre si. Dessa forma, estes objetos e este espaço de trabalho vão ter importantes papéis no desempenho da equipe como um todo, uma vez que boa parte da cooperação se dará através deste ambiente e destes objetos. Além disso, é importante que os membros de um grupo percebam a informação de percepção de modo natural, sem perder o tempo em que poderiam estar

cooperando para buscar esta informação. Assim, idealmente, esta informação deveria ser colocada dentro deste espaço e destes objetos, pois é neles que o trabalho é realizado.

A percepção do que está ocorrendo neste espaço de trabalho compartilhado é chamada de *workspace awareness* e pode ser comparada à percepção que as pessoas ao redor de uma mesa em uma reunião têm umas das outras e do trabalho que está sendo feito. A *workspace awareness* é a compreensão da interação de outra pessoa com um espaço de trabalho compartilhado neste momento, envolvendo o conhecimento sobre onde este alguém está trabalhando, o que está fazendo e o que vai fazer a seguir [GUT 99]. Saber o que está ocorrendo neste exato momento no ambiente de trabalho compartilhado é vital, quando se tem os membros do grupo trabalhando ao mesmo tempo. Em outras palavras, quando se tem um grupo de pessoas trabalhando simultaneamente em um ambiente síncrono, faz-se necessário fornecer suporte à *workspace awareness*. Esta age, segundo Gutwin e Greenberg [GUT 99], como uma base comum, um conhecimento mútuo, do qual as pessoas tiram proveito para aumentar a eficiência da comunicação.

A *workspace awareness* trata do conhecimento do que está acontecendo no espaço de trabalho compartilhado no momento atual e é um elemento natural dentro de ambientes físicos de trabalho. Por exemplo, quando em uma reunião face a face, em um grupo de trabalho real, os participantes, mesmo que inconscientemente, precisam saber o que os colegas estão fazendo ao seu redor para que possam coordenar melhor suas próprias atividades. Para Gutwin e Greenberg [GUT 98],[GUT 99], um participante, em uma situação como esta, pode ver todo o *workspace* físico e quem estiver nele, mesmo se alguém estiver envolvido em tarefas individuais. Nestes ambientes, as pessoas interagem com artefatos físicos, manipulando-os, o que também dá aos demais informações sobre a natureza e o progresso das atividades.

Assim, ao se considerar ambientes físicos, três fontes de informação são produzidas: *consequential communication*, *feedthrough* e *intentional communication*. O primeiro, relaciona-se aos movimentos característicos de uma ação, os quais comunicam seu caráter e conteúdo aos demais membros do grupo. Trata-se do próprio mecanismo de ver e ouvir outras pessoas através do espaço de trabalho, onde a transferência da informação vem como conseqüência das atividades dos outros. O *feedthrough* refere-se ao *feedback* produzido quando os artefatos são manipulados, o qual fornece aos demais membros do grupo informações sobre esta manipulação. A *intentional communication* consiste nas informações obtidas através de gestos e conversação entre os membros. Estes são conhecimentos naturais e inerentes em ambientes físicos, mas em um *workspace* virtual, não o são. Nestes ambientes, os próprios recursos e a flexibilidade dos dispositivos de entrada e saída são menores se comparados aos ambientes físicos, mas, em compensação, existem menos restrições sobre as interações no s ambientes "virtuais" do que em *workspaces* reais [GUT 99], [GUT 98].

Dessa forma, toda a passagem das informações de *workspace awareness* para os membros de um grupo em um ambiente de trabalho virtual é de inteira responsabilidade do projetista do sistema e este pode fazer uso da liberdade que os ambientes virtuais oferecem para buscar maneiras mais adequadas de fornecer percepção. O importante é explorar estas possibilidades de modo a melhor fornecer a percepção do que está ocorrendo no ambiente de trabalho síncrono para o grupo. Estas informações são condição vital e necessária para este tipo de ambiente, sem as quais o trabalho dos membros pode ser prejudicado.

Já em ambientes assíncronos, como não há a obrigatoriedade dos membros estarem trabalhando simultaneamente, a percepção de "onde" está ocorrendo as ações é diferente desta percepção em ambientes síncronos, onde o foco encontra-se na percepção do espaço de trabalho (*workspace awareness*). O foco maior de atenção em ambientes assíncronos são os objetos compartilhados. É através deles que a comunicação entre os membros se dará, através de sua manipulação e de seu histórico de manipulação. Este histórico vai fornecer informações sobre o que houve e está acontecendo dentro do trabalho em grupo, mostrando aos membros o que foi feito sobre os objetos compartilhados, criando assim um contexto para as atividades de cada membro. Segundo Dourish [DOU 97], o artefato compartilhado é essencialmente o único espaço compartilhado disponível aos participantes nestes ambientes e representa também a informação chave na colaboração assíncrona. Assim, o projetista de um sistema assíncrono, ao desenvolver o suporte à percepção, deve priorizar o fornecimento das informações de percepção através dos objetos compartilhados pelos membros e não puramente através do próprio espaço de trabalho, já que em ambientes assíncronos não há nenhuma garantia de que estes membros estarão trabalhando em algum momento simultaneamente sobre este espaço.

Em ambos os casos, ambientes síncronos e assíncronos, como é de inteira responsabilidade dos projetistas o desenvolvimento de um suporte eficaz, tanto para *workspace awareness* quanto para a percepção através dos objetos compartilhados, é importante que estes projetistas considerem qual a melhor metáfora a ser usada nestas representações. Podem ser usadas tanto metáforas do próprio ambiente de trabalho real, como um escritório, ideais para sistemas síncronos, com elementos presentes no dia-a-dia dos participantes, como uma mesa de reunião, um quadro de avisos ou um *whiteboard*, os quais podem minimizar a possibilidade de más interpretações [VEE 97], quanto uma metáfora do próprio tipo de sistema, relacionando a ferramenta de *groupware* com versões mono-usuários do mesmo tipo de sistema, como, por exemplo, editores cooperativos que se assemelham a editores mono-usuários, metáforas mais adequadas a ambientes assíncronos.

A metáfora utilizada afeta o modo como as informações podem ser capturadas pelos participantes, havendo a necessidade de enriquecê-la adequadamente com informações de *awareness*. Este enriquecimento deve agir no sentido de permitir aos participantes perceberem mais claramente as atividades do grupo, enfatizando os aspectos de cooperação, fornecendo-lhes a sensação de estarem realmente trabalhando em grupo. Esta preocupação é mais marcante quando na presença de metáforas que usam o tipo de sistema como base. Estas devem ser realmente enriquecidas para não permitir que os usuários se prendam à analogia com um sistema mono-usuário e percam o contato com o grupo. Por exemplo, uma metáfora de *desktop*, empregada em muitos sistemas tradicionalmente utilizados pelos usuários de forma isolada, como os sistemas operacionais Microsoft Windows© e IBM OS/2©, precisa ter um enriquecimento com informações de percepção sobre os colegas, se alguém está presente (*on-line*), que atividades foram e vêm sendo realizadas sobre os objetos compartilhados e seus prazos de conclusão. Tudo isso para o usuário, acostumado a trabalhar isolado neste tipo de metáfora, não se sinta assim e perceba que existe um grupo trabalhando com ele, mesmo que não simultaneamente.

O mesmo já não ocorre com tanta força em ambientes síncronos, onde a analogia com o ambiente real de trabalho, juntamente com a própria presença dos colegas no ambiente virtual, auxiliam a manter este contato. Nestes ambientes, o uso de uma metáfora de escritório virtual, com elementos como salas, mesas e quadros, lembram

naturalmente o usuário de que ele é parte de uma equipe, uma vez que estes elementos são também naturais em ambientes reais.

Sendo assim, a tabela abaixo (Tab. 3.3) resume as considerações colocadas acima quanto à questão “onde”, onde as informações são geradas e apresentadas, relacionando-se ao espaço de trabalho e à metáfora utilizada neste espaço.

TABELA 3.3 - A questão "onde" para a percepção

		Ambientes Síncronos	***Ambientes Assíncronos***
Onde	*Espaço*	*Workspace awareness*	Objetos compartilhados
	Metáfora	Escritório	Sistema

3.2.4 Como

A próxima questão sobre percepção é "como", ou seja, como as informações são apresentadas aos usuários. Esta é uma questão intrinsecamente de interface, a qual constitui um ponto marcante para a percepção. A interface com o usuário é a responsável por fornecer aos membros do grupo as informações de *awareness* e deve apresentar estas informações de forma resumida, a fim de evitar a sobrecarga dos membros e de modo a permitir uma rápida assimilação destas pelos mesmos. Os membros não podem se sentir soterrados pelas informações de percepção, nem terem omitidas informações importantes. A informação de *awareness* deve ser de natureza passiva, ela deve surgir direto das atividades de cada pessoa, ao invés de ser gerenciada ou procurada explicitamente [DOU 97]. Uma interface mal projetada pode tanto pecar pela omissão quanto pelo excesso. O excesso na quantidade de informações relativas à percepção pode causar sobrecarga cognitiva aos usuários. Estes poderão ficar tão envolvidos com a manipulação da informação vinda do mecanismo de *awareness* que poderão não conseguir cooperar suficientemente com o grupo, como ilustra a Fig. 3.3 abaixo. Nesta figura, o membro à esquerda é soterrado por gráficos, notas e *news* que não consegue chegar até o grupo que coopera à direita.

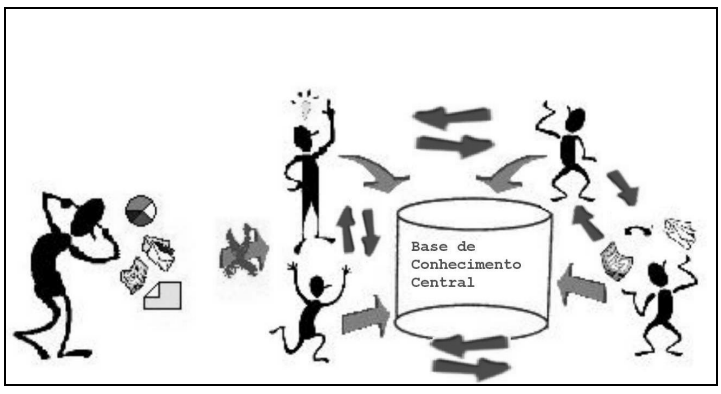


FIGURA 3.3 - Exemplo de excesso de informações de percepção sobre um membro

Portanto, para fazer o projeto de uma interface balanceada, deve-se procurar por elementos de interface adequados, que apresentem as informações de forma reduzida, sem soterrar o grupo com dados de pouca utilidade, nem deixar de apresentar aquelas importantes. Estes elementos podem ser ícones ou cores associados a informações específicas, como papéis, participantes e atividades. Podem ser gráficos representativos do progresso do trabalho de todo o grupo, de um membro ou de uma atividade; resumos textuais sobre o trabalho, como resumo de "logs" de atividades; ou ainda uma combinação de elementos como estes. O importante é que estes elementos apresentem adequadamente as informações de percepção aos membros, de forma resumida, sem sobrecarga, nem perda de conteúdo significativo. Para tanto, estes elementos deverão fazer uma filtragem ou um agrupamento das informações, mostrando apenas aquilo que provavelmente será mais útil e interessante a cada participante.

Estes processos de filtragem e agrupamento podem utilizar vários critérios, como as atividades envolvidas, as preferências pessoais e o papel desempenhado pelo próprio membro. As preferências pessoais e os papéis desempenhados são critérios determinantes para a delimitação de que informações serão de interesse de cada membro. As preferências pessoais vão incluir dentro do campo da percepção os interesses particulares de cada indivíduo, enquanto os papéis, conforme discutido na questão "o que", são vitais e devem influenciar na escolha das informações para a percepção. Esta influência vai afetar a escolha das informações, pois de acordo com o papel desempenhado por cada membro, certas informações serão essenciais para o seu melhor desempenho, como é o caso da coordenação, também discutido anteriormente. Aliando-se, então, as preferências pessoais e os papéis desempenhados, cada membro poderá ter uma percepção de atividades diferentes de seus colegas. Outros exemplos de filtragem e agrupamento de informações podem ser obtidos na literatura, como em Hudson e Smith [HUD 96], onde alguns exemplos de filtros para as informações de presença e atividades dos colegas são dados. Posteriormente, durante a última das questões sobre percepção, este ponto específico da quantidade de informação retornará a discussão em maiores detalhes. Aqui, a preocupação é tão somente a interface para representar as informações.

Além desta preocupação geral quanto à qualidade da interface, há também a questão da apresentação dos objetos e do espaço de trabalho compartilhado. Como já foi mencionado anteriormente, quando se trabalha em grupo, trabalha-se sobre um conjunto de objetos colocados dentro de um espaço de trabalho compartilhado pelos membros do grupo. É importante que estes membros saibam o que acontece sobre cada um dos objetos ou sobre aqueles que mais lhes interessam. Além disso, se todos os membros vêem os objetos da mesma forma ou se há diferenças nas visões de cada um afeta sensivelmente a percepção do grupo com relação a estes objetos, pois manter uma mesma visão para todo o grupo significa que os membros vão ter disponíveis as mesmas informações, logo, vão ter a mesma percepção destes objetos compartilhados e do que aconteceu e acontece sobre eles. O mesmo vale para a visão que os membros têm do espaço de trabalho como um todo, pois visões diferentes vão implicar em informações diferentes para cada membro, logo, percepções diferentes (quando a representação diverge, o contexto pode se perder [GUT 98]).

Com isso, forma-se uma disputa entre as necessidades do grupo e a praticidade para o usuário. Segundo Gutwin e Greenberg [GUT 98], os usuários de um *groupware*, mesmo durante o trabalho síncrono, agem tanto como indivíduos quanto como membros do grupo. Como indivíduos, precisam de meios poderosos e flexíveis para interagir com o *workspace* e seus objetos, enquanto o grupo precisa de informação sobre os outros

para manter a percepção. Diferentes representações do espaço de trabalho e seus objetos podem tornar as tarefas individuais mais fáceis, mas também podem restringir a comunicação sobre os mesmos objetos, sendo mais fácil manter a percepção do espaço de trabalho quando uma mesma representação é mantida.

Dessa forma, o projetista de um *groupware* deve decidir entre beneficiar mais as atividades individuais, feitas pelos membros do grupo de forma isolada, ou priorizar a percepção para as atividades coletivas do grupo. Esta é uma decisão que depende muito do tipo de sistema tratado, se é um sistema síncrono ou assíncrono. Em sistemas síncronos, o trabalho é feito pelo grupo simultaneamente, os membros estão trabalhando ao mesmo tempo sobre o espaço e os objetos de trabalho compartilhados. Isto faz com que a percepção mais uniforme deste espaço de trabalho e dos objetos que o compõe seja mais importante do que a flexibilidade para o usuário isolado. Isto porque o contexto das atividades realizadas pelo grupo deve ser mantido e este contexto é representado no espaço de trabalho compartilhado. É importante, nestes ambientes, ter a oportunidade de ver onde seu colega está trabalhando agora e o que está fazendo em detalhes. Assim, costuma-se usar, nestes casos, interfaces com maior acoplamento, como interfaces WYSIWIS _ *What You See Is What I See*, e WYSIWIS relaxadas, ou ainda recursos como múltiplas visões.

Nas interfaces WYSIWIS, todos os membros ativos vêem a mesma imagem, se um se desloca no espaço compartilhado, o mesmo ocorre na tela dos demais (por exemplo, um *page down* na janela de um usuário terá o mesmo efeito na janela dos demais). Dessa forma, garante-se que o contexto das atividades será mantido, já que os membros conectados estarão vendo exatamente a mesma coisa. Entretanto, esta é uma interface muito restritiva e pode, algumas vezes, prejudicar o andamento do trabalho por tolher demais a liberdade dos membros como indivíduos. Para estes casos, talvez a solução mais adequada seja uma interface WYSIWIS relaxada, especialmente quanto à navegação dos participantes, como afirmam Smith e O'Brien [SMI 98]. Para estes autores [SMI 98], as interfaces WYSIWIS são muito inflexíveis e devem ser relaxadas para melhor acomodar as interações, e a noção de interfaces WYSIWIS relaxadas fornecem a cada indivíduo a habilidade de configurar a sua interface compartilhada para melhor se adaptar as suas necessidades de trabalho.

Todavia, as interfaces WYSIWIS relaxadas, quando em grandes *workspaces*, seguidamente falham ao trazer informações de *workspace awareness* [GUT 96], o que significa que os usuários podem até mesmo perder algumas informações, como não perceber que parte de um grande diagrama está sendo removida, mas em compensação, há uma maior liberdade para os membros realizarem suas tarefas. Além disso, recursos como visão "olho de peixe" (*fisheye*), *telepointers* e *multi-users scrollbars* podem dar aos participantes idéia das atividades dos colegas, sem perder muito de sua liberdade de atuação individual. Estes recursos de interface, normalmente chamados de *awareness widgets*, são elementos de interface especificamente projetados para o fornecimento de *awareness* [PRA 99]. Vários exemplos deste tipo de recursos podem ser encontrados em Gutwin e Greenberg [GUT 98], e uma interessante avaliação prática de alguns destes recursos foi realizada por Gutwin e Roseman [GUT 96]. A Fig. 3.4 abaixo traz um exemplo de *awareness widget* chamado "*Gestalt Viewer*". Projetado para ser usado durante a edição cooperativa de documentos, o Gestalt apresenta uma miniatura do documento que está sendo trabalhado sobreposta com retângulos coloridos, indicando onde cada participante está trabalhando [ROS 96].

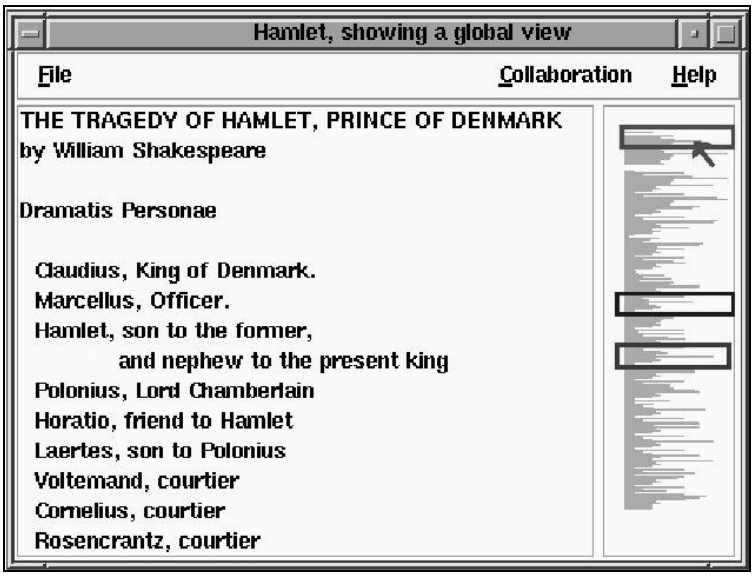


FIGURA 3.4 - *Awareness widget Gestalt viewer* [ROS 96]

Além das interfaces WYSIWIS, os sistemas síncronos podem ainda contar com o recurso de múltiplas visões, onde um participante pode escolher ver o ambiente compartilhado e seus objetos de diversas maneiras. Estas representações são um recurso interessante para os participantes, especialmente junto a interfaces WYSIWIS relaxadas, onde esta é uma opção viável para melhor manter a percepção dos colegas. Por exemplo, para o coordenador de um grupo pode ser interessante, por vezes, ter exatamente a visão que outro participante está tendo, como se olhasse por cima de seus ombros, ou ainda, um participante pode manter uma visão geral do *workspace*, com a posição e as atividades de cada colega, enquanto mantém uma outra visão mais apropriada a sua atividade individual.

Em ambientes assíncronos, os usuários não precisam necessariamente trabalhar ao mesmo tempo, pode haver um intervalo de tempo entre as atividades de cada membro do grupo. Sendo assim, não há sentido algum em se utilizar interfaces WYSIWIS puras, que enfocam as ações dos colegas no momento presente, pois não há garantias de que eles estarão trabalhando ao mesmo tempo. Todavia, a presença de mais de um usuário no sistema pode ser encarada como uma boa oportunidade para cooperação, devendo ser explorada. Para estes casos, a aproximação mais adequada são as interfaces WYSIWIS relaxadas, dão liberdade de ação a cada participante, sem deixar de mostrar o que os demais colegas ativos no momento estão fazendo. Outra possibilidade é também unir as vantagens de uma visão WYSIWIS relaxada com a idéia de múltiplas visões, fornecendo aos participantes a oportunidade de escolher entre uma visão mais adequada a sua atividade individual e outra visão que favoreça mais a sua colaboração com os possíveis colegas conectados.

Uma última possibilidade é a utilização de interfaces totalmente desacopladas, as quais não fornecem a seus usuários informações sobre as atividades dos colegas ou sobre o espaço de trabalho compartilhado. Este caso é mais utilizado quando o centro da cooperação está no conteúdo das atividades, e informações como a posição de um colega no trabalho não é tão importante quanto, por exemplo, saber quais as atividades sob a sua responsabilidade. Todavia, vale ressaltar que isto não significa que este tipo de interface não vá trazer nenhuma informação de percepção. Neste caso, a interface simplesmente não está acoplada às interfaces vistas pelos colegas, podendo cada

membro adaptar a sua interface as suas necessidades individuais sem que os demais sintam ou sejam notificados disto, o que não impede esta mesma interface de incluir elementos para percepção, tais como um medidor de participação dos usuários, mostrando quem tem trabalhado mais, ou um gráfico mostrando as atividades já concluídas e as que ainda faltam, úteis para a coordenação do grupo; ou um medidor de qual percentual da atividade já foi concluído.

TABELA 3.4 - Percepção de "Como".

		Ambientes Síncronos	***Ambientes Assíncronos***
Como	*Interface*	WYSIWIS	WYSIWIS relaxado
		WYSIWIS relaxado	Múltiplas Visões
		Múltiplas Visões	Desacopladas
	Balanceamento	Filtragem	
		Agrupamento	

3.2.5 Quem

A penúltima questão a ser feita é “quem” e se refere ao conhecimento de quem está trabalhando e quem está atento no grupo no momento. Este é um conhecimento importante para o bom andamento das atividades no grupo, sendo natural em ambientes reais, como, por exemplo, uma reunião face a face, onde basta aos participantes olharem ao redor para saber quem está presente e atento. Ele age como facilitador da cooperação, à medida que estimula a interação e a comunicação informal entre os membros. Segundo Smith [SMI 98], boa parte do trabalho que tem lugar em um ambiente de escritório é baseada na comunicação informal, a qual, em muitos casos, permite às pessoas ficarem conscientes das atividades dos colegas e fornece também um contexto para as atividades relacionadas ao trabalho.

A noção de presença dos outros participantes, para Araújo *et al.* [ARA 97], é o tipo mais comum de informação oferecida aos membros e está relacionada a mecanismos de notificação dos membros que estão ativos a cada momento. Estes mecanismos de notificação são vitais em ambientes síncronos, cuja essência da cooperação encontra-se nas interações síncronas entre os participantes. Nestes ambientes, os participantes precisam (obrigação) estar conscientes da presença dos demais para que o trabalho prossiga e obtenha resultados satisfatórios, pois toda a essência do trabalho está nas atividades conjuntas e simultâneas dos membros. Sendo assim, torna-se inviável realizar uma tarefa simultaneamente com um grupo de pessoas sem saber quem é este grupo.

Em ambientes síncronos, a noção de presença também está ligada à questão da autoria das atividades, uma vez que saber quem está ativo no momento facilita a identificação de quem fez o que no ambiente, onde cada membro está executando suas tarefas, simultaneamente aos demais. Dessa forma, a percepção da presença dos membros está diretamente ligada à *worksapace awareness* no tocante ao conhecimento de quem está ativo no espaço de trabalho compartilhado, pois esta é importante item para a percepção do que ocorre no espaço de trabalho. Somente de posse desta percepção da presença dos colegas é possível montar o contexto completo de quem está trabalhando no *workspace*, e, a partir daí, o que está fazendo e onde.

Já em ambientes assíncronos, a noção de presença comporta-se mais como uma oportunidade extra para a cooperação entre os membros e não uma necessidade. Isto porque não há uma obrigatoriedade na presença de todos no sistema para que o trabalho em grupo prossiga, como é o caso em sistemas síncronos. Sendo assim, a presença de mais de um membro ao mesmo tempo no sistema deve ser encarada como uma coincidência e, como tal, representa uma oportunidade a ser explorada pelo grupo. Quando mais de um membro está ativo ao mesmo tempo, eles podem trocar idéias, experiências e dúvidas, aproveitando a situação para enriquecer o conhecimento deles do grupo, a própria cooperação e, consequentemente, o trabalho como um todo.

A Fig. 3.5 abaixo, retirada de [GRE 98], mostra um exemplo de mecanismo implementando pelo GroupKit [GRE 98], [ROS 96], [ROS 97], que apresenta uma lista com os participantes presentes e ainda permite ao usuário consultar maiores informações sobre os colegas presentes, como foto, telefone e correio eletrônico. Fornecer informações sobre quem são os colegas é muito importante para o fortalecimento da noção de grupo, tanto em ambientes síncronos quanto assíncronos. Esse conhecimento cria um senso maior de comunidade, à medida que os membros ficam sabendo mais sobre seus colegas, e juntamente com mecanismos de comunicação, discutidos a seguir, vão reforçar a coesão entre estes membros.

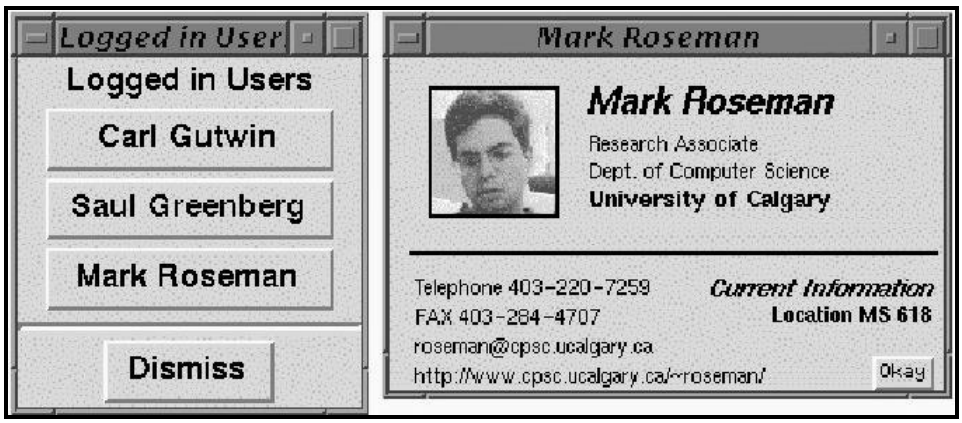


FIGURA 3.5 - Informações detalhadas sobre um participante [GRE 98]

A noção da presença de quem está conectado no momento envolve também saber se este alguém está ou não atento ao sistema, pois estando os membros geograficamente distantes, a mera presença de alguém no sistema não garante que ele realmente esteja atento. Ele pode não estar em frente ao computador, estar realizando uma atividade individual, ou qualquer outra tarefa fora do sistema. De posse deste conhecimento, é possível a um membro iniciar uma conversa, trocar idéias, pedir auxílio a seus colegas ou até mesmo resolver os possíveis conflitos que podem surgir. Neste sentido, a noção de presença atua em um *groupware*, diretamente na noção de percepção informal apresentada por Greenberg (o senso geral de quem está ao redor e o que estão fazendo) [GRE 96], e graças a esta noção que a comunicação informal entre os membros de um grupo tem seu lugar dentro de sistemas cooperativos suportados por computador. Através das ferramentas de comunicação, estes membros podem explorar estas possibilidades de comunicação, permitindo tornar suas relações mais pessoais e interativas e não tão frias e formais.

Estas ferramentas de comunicação devem ser adaptadas ao tipo de sistema onde estão inseridas. Por exemplo, em sistemas síncronos, elas devem priorizar a comunicação síncrona entre os participantes, com ferramentas como *chat, talks, whiteboards* e outras ferramentas no estilo conferência, aproveitando-se do fato destes participantes estarem sempre trabalhando simultaneamente. Já em ambientes assíncronos, é mais importante que ferramentas se adaptem ao fato dos membros não estarem todos trabalhando ao mesmo tempo, destacando-se as ferramentas assíncronas, como correio eletrônico, quadro de avisos e notas. O uso de ferramentas síncronas, como *chats*, neste caso, ficam sujeitas à eventual presença de mais de um membro do grupo ao mesmo tempo, sendo, portanto, bastante limitadas, já que não há garantias de tempo nestes ambientes. Com isso, a disponibilidade de ferramentas assíncronas, como o *email*, é praticamente obrigatória para que os membros em um ambiente assíncrono tenham a oportunidade de manter o nível de comunicação informal com seus colegas, tão importante para a colaboração, ficando as ferramentas síncronas somente como um recurso extra para esta comunicação.

Sendo assim, a Tabela 5 abaixo resume a questão "quem", com as características da percepção expostas acima.

TABELA 3.5 - A questão "quem" para percepção

		Ambientes Síncronos	***Ambientes Assíncronos***
Quem	*Presença*	obrigatoriedade	oportunidade
	Ferramentas de Comunicação	Essencialmente ferramentas síncronas	Essencialmente ferramentas assíncronas

3.2.6 Quanto

A última das questões a se fazer sobre percepção é "quanto" e destaca uma característica essencial que deve ser observada no suporte à percepção: quanto de informação é o suficiente. A questão gira em torno de qual a quantidade ideal de informações que deve ser apresentada ao usuário, a fim de lhe prover percepção sobre o grupo e suas atividades. É o somatório de todas as informações geradas pelas demais questões, o qual deve fornecer informações suficientes, sem sobrecarregar o usuário.

Esta preocupação de fornecer a quantidade certa de informações para a percepção, sem causar nem a insuficiência, que poderia deixar os usuários perdidos, sem um contexto para as atividades em grupo, nem a sobrecarga, que pode prejudicar as atividades dos participantes à medida que interrompe o curso do trabalho e exige de cada um atenção excessiva para filtrar as informações de seu interesse, é uma constante no desenvolvimento de um suporte eficaz à percepção, afetando praticamente todas as questões anteriormente discutidas. Por exemplo, com relação à questão "onde", ao mencionar a *workspace awareness*, deve-se ponderar que tornar as ações dos colegas mais perceptíveis ajuda a manter a *awareness*, mas quanto mais informação visual é mostrada, maior o risco de distrair os indivíduos. Assim, nem todos os eventos podem ser mostrados no *workspace* dos colegas da mesma forma que aparecem na tela do usuário local, algumas ações podem se reduzidas, enquanto outras menores, podem ser ampliadas [GUT 98].

Já quando se trata da questão “quem”, deve-se considerar quanta informação sobre os colegas deve ser apresentada. Em geral, quanto mais informação é transmitida sobre as ações de alguém, maior o potencial para a percepção, mas, ao mesmo tempo, maior é o potencial de se violar a privacidade deste alguém (considere, por exemplo, um usuário com escritório em casa, o uso de recursos como uma *webcam* podem fornecer inúmeras informações sobre suas atividades, mas também pode invadir a privacidade de seu lar). E quanto mais informações se recebe sobre os outros, maior percepção sobre eles é possível e maior a possibilidade que isso interrompa as atividades ou consuma muitos recursos [HUD 96].

O fato é que os usuários não querem ser inundados com informações de *awareness* para cada evento que ocorre [MAR 97]. Segundo Prakash et. al. [PRA 99], uma grande quantidade de dados pode estar disponível para os membros do grupo e estes dados compartilhados podem sofrer alterações freqüentemente, através das atualizações por membros ou fontes externas, mas os usuários não querem necessariamente serem providos de todos estes dados, nem notificados de todas as atualizações feitas. Isto leva os desenvolvedores de um *groupware* a se questionarem quanta informação é suficiente para a percepção. Quantificar uma solução para isso numericamente é totalmente inviável, devido à complexidade envolvida, já que esta resposta depende de fatores como o ambiente onde estão inseridos os participantes, o tipo de atividade realizada, o papel e os interesses de cada membro.

O ponto chave para esta questão é que não é interessante nem omitir todas as informações, tampouco apresentá-las todas. É necessário buscar um meio termo entre a insuficiência e a sobrecarrega de informações de modo a se adequar às necessidades do ambiente a que se propõe o *groupware* e aos interesses dos membros. Uma solução para isto é extrapolar a idéia de filtragem e agrupamento das informações para além da interface, para além das dimensões do “como” apresentar as informações. É colocar esta mesma idéia de filtrar ou agrupar as informações na escolha das informações (“o que” - quais as informações que são do interesse pessoal de cada membro) e na escolha do momento da apresentação (“quando” - quando o usuário quer e não quer ser notificado das informações de percepção). É dar vazão à opinião e aos interesses dos usuários, permitir que eles possam escolher os eventos que lhe são interessantes e quando é mais oportuno recebê-los. Esta possibilidade significa permitir aos próprios membros colocarem critérios particulares para a filtragem e agrupamento das informações, montando espécies de *profiles* próprios, baseados em seus interesses e aspirações particulares. Por exemplo, um usuário poderia registrar seu interesse em objetos no sistema e ser notificado de qualquer tentativa de acesso a estes objetos [MAR 97].

Dessa forma, a questão quanto afeta a todas as demais, o que, quando, como, onde e quem, caracterizando-se como uma dimensão paralela as demais, afetando não só a cada uma delas de forma isolada, mas também o somatório, ao conjunto de todas as informações extraídas das outras cinco questões, indicadas na Fig. 3.6 abaixo pelo símbolo “$\Sigma_{\text{informações}}$”. Este conjunto não pode ficar em nenhuma das extremidades do “quanto” indicadas na Fig. 3.6 abaixo, nem na insuficiência, nem na sobrecarga.

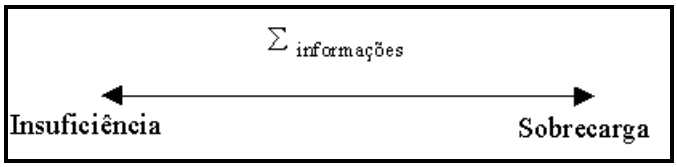


FIGURA 3.6 - A dimensão "quanto"

Uma outra forma de visualizar esta questão é vê-la como função da densidade de informação. Imaginando que cada ponto discutido até agora, como atividades em micro-nível, eventos no passado e interface WYSIWIS, possa variar entre não fornecer informação alguma (equivalente a não ter suporte a esta característica), fornecer um conjunto de informações pequeno e fornecer um vasto conjunto de informações, conforme indica a variação de cor apresentada na Fig. 3.7 abaixo, a dimensão "quanto" representaria o conjunto de todas estas informações, o qual poderia ser um conjunto mais ou menos denso. Esta visão mantém a mesma idéia central da visão apresentada antes (Fig. 3.6), de que a questão quanto é uma dimensão paralela as demais, definida por todo o conjunto de informações indicados pelas outras questões e cujo ideal de um *groupware* seria o de alcançar um ponto entre um conjunto muito denso e um muito pouco denso destas informações. Ou seja, a idéia central permanece a de chegar a um ponto intermediário, com informações suficientes para que seja mantido o contexto do grupo sem que o trabalho deste seja prejudicado, de acordo, é claro, com os interesses desse grupo e do próprio *groupware*.

FIGURA 3.7 - Variação na densidade da informação

3.2.7 O esquema de classificação

Discutidas as seis questões que delimitam as características consideradas mais importantes para percepção, resta agora apresentar na Fig. 3.8 o esquema de classificação que se formou desta discussão, o qual serve como forma de classificação para os mecanismos de suporte à percepção que serão analisados na próxima seção. Pode-se observar na Fig. 3.8 que a questão "quanto", comentada logo acima, é apresentada como uma dimensão paralela as demais, já que ela exerce forte influência sobre cada uma das outras questões e sobre o conjunto formado por todas as informações apresentadas pelas demais. Além disso, este quadro, ao resumir tudo que foi discutido até agora, constrói uma visão geral do que é necessário observar e optar durante o desenvolvimento de um suporte eficaz à percepção.

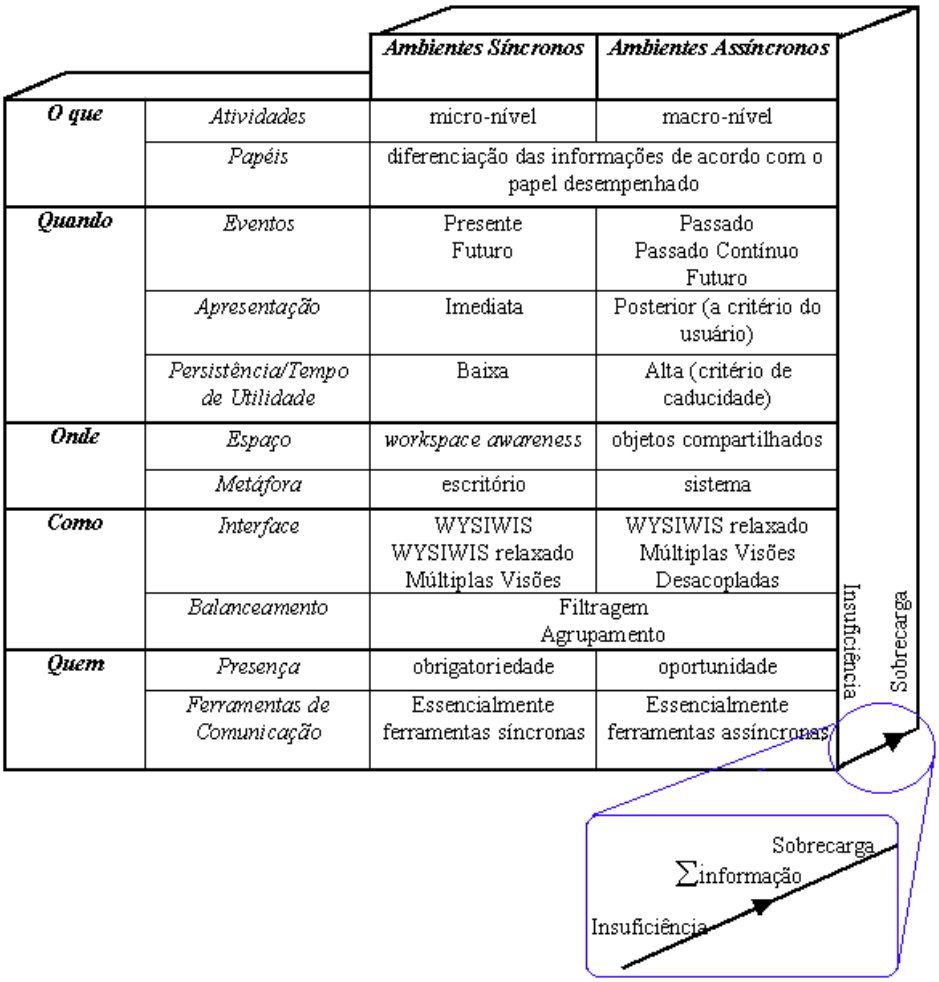


FIGURA 3.8 - Esquema de classificação para percepção

3.3 Mecanismo para suporte à percepção

Conforme dito anteriormente, a percepção é fornecida aos usuários por um conjunto de mecanismos. Estes mecanismos transformam interações irregulares em interações consistentes e perceptíveis, permitindo aos membros do grupo manterem-se atualizados sobre eventos importantes, contribuindo para que suas atividades sejam realizadas de modo consistente e eficaz [ARA 97]. Para tanto, estes mecanismos deveriam observar algumas das características apresentadas no *framework* acima, de acordo com o tipo de percepção que desejam fornecer, e a partir deste ponto escolher uma ou mais abordagens para sua implementação. Dias [DIA 98] apresenta quatro abordagens para a construção destes mecanismos de suporte:

- Recursos de interface (*awareness widgets*): os *awareness widget*, como *telepointers*, *multi-user scrollbars*, mecanismos de radar e visão olho-de-peixe, podem ser facilmente integrados ao próprio ambiente de trabalho, facilitando a absorção das informações de percepção pelos seus usuários;

- Consulta e navegação pela memória de grupo, com a procura de informações sobre as interações e as atividades de cada membro;
- Anotações: o uso de anotações pode ser um mecanismo eficiente de interação entre os membros, registrando idéias, sugestões e comentários, especialmente útil quando os membros não estão todos trabalhando em um mesmo intervalo de tempo (ambientes assíncronos);
- Notificações: o uso de um mecanismo de notificação permite a cada membro tomar conhecimento de eventos ocorridos durante a interação com o grupo.

Estas abordagens podem ser combinadas e ampliadas em um único mecanismo de suporte à percepção, podendo assim se adequar às necessidades do sistema, pois a mais importante função destes mecanismos é justamente fornecer aos membros do grupo informações sobre o andamento do trabalho que lhes afeta.

Analisou-se diversos mecanismos encontrados dentro de várias ferramentas relatadas na bibliografia e o resultado desta análise é apresentado a seguir, da seguinte forma: inicialmente, são comentadas todas as ferramentas. Para cada uma é relacionado um símbolo para, ao final da seção, distribuir estes símbolos coloridos na classificação apresentada na seção anterior, a fim de mostrar como a percepção dada pelos mecanismos implementados por cada uma destas ferramentas encaixa-se na distribuição geral. A cada símbolo também pode ser associado um sinal de menos (“-“) para indicar que o suporte à característica não está sendo completo ou adequado (poucas informações a seu respeito são fornecidas). Todavia, não foi analisada dentro desta seção a questão “quanto”, uma vez ser esta uma questão delicada, que exige um estudo mais aprofundado, com avaliações mais detalhadas, levando-se em conta fatores como os objetivos propostos por cada mecanismo e o tipo de usuário e situação considerados alvos, o que conduz à necessidade de cuidadosos testes práticos, o que não foi realizado aqui em todos os casos.

3.3.1 COPSE / CUTE (α)

O COPSE (*Collaborative Project Support Environment)* é um ambiente para o desenvolvimento cooperativo de software e inclui um ambiente de desenvolvimento que permite a integração de ferramentas para as várias etapas da construção cooperativa de um *software*, a chamada infra-estrutura, e um *framework* para a construção de aplicações *groupware* independentes do ambiente, mas que podem ser, durante ou após seu desenvolvimento, acopladas a este ambiente. O CUTE (*Collaborative UML Technique Editor)* é um editor com suporte a aspectos de cooperação na modelagem de *softwares* orientados a objetos, focando as atividades de diagramação deste. O CUTE foi construído sobre o *framework* do COPSE e permite a edição de diagramas de classes simples, usando elementos de classes e de relacionamentos, com notação da UML para a visualização do diagrama [DIA 98], [DIA 97].

O COPSE considera que várias são as atividades realizadas no decorrer do desenvolvimento de um *software* e cada uma destas atividades teria ferramentas associadas adequadas à situação. Assim, a idéia do ambiente COPSE é interligar estas ferramentas, através de sua infra-estrutura, permitindo o desenvolvimento integrado do

software. Estas ferramentas poderiam ser construídas, fazendo uso do próprio *framework* COPSE, o qual pode ser utilizado também para o desenvolvimento de ferramentas de *groupware* não acopladas à infra-estrutura COPSE, como o CUTE.

O CUTE foi todo construído sobre o *framework* do COPSE e suporta o trabalho síncrono em equipe, possuindo uma arquitetura cliente/servidor (modelo também adotado por todo o COPSE) e um protocolo de comunicação TCP/IP, baseado no envio de objetos serializados (tanto o COPSE quanto o CUTE foram todos construídos em linguagem Java). Estes objetos são gerenciados pelo servidor que também os armazena, enquanto o cliente fica responsável pela apresentação visual, mantendo somente uma cópia consistente destes. Cada processo servidor trabalha sobre um mesmo projeto e é capaz de atender a diversos clientes sobre este projeto, fazendo com que todos os clientes que estiverem conectados a ele trabalhem sobre um mesmo diagrama de classes. Nenhuma restrição de segurança sobre o diagrama de classes é realizada. Não é solicitada nenhuma forma de senha e todos os usuários conectados ao servidor podem alterar o diagrama, não havendo divisão de papéis entre os membros. A razão dessa ausência é o fato do diagramador não estar integrado ao ambiente do COPSE, utilizando recursos somente do *framework*.

No ambiente COPSE, o servidor controla todo o processo de produção de *software* e as atividades que o compõe, bem como os membros do grupo, suas permissões e os produtos gerados. Os clientes se conectam a este servidor, solicitando a realização de uma atividade dentro do processo. O servidor verifica o processo para ver se a atividade pode ser realizada e, a partir de então, dispara o servidor específico para aquela atividade. Cada atividade do processo de produção do *software* tem relacionada uma ferramenta integrada ao ambiente responsável por seu desenvolvimento, sendo o servidor dessa ferramenta disparado pelo próprio servidor do ambiente [DIA 98].

Tanto o COPSE quanto o próprio CUTE mantém um suporte restrito à percepção, essencialmente centrado nas necessidades do trabalho síncrono. O COPSE inclui apenas a descrição de uma memória de grupo, que pode ser útil para o suporte à percepção de eventos no passado, mas no CUTE há somente a percepção de eventos presentes, com apresentação imediata, não havendo persistência para as informações de percepção.

O CUTE mantém a noção de presença no *workspace*, graças ao recurso de "*user list*" e mantém ferramentas de comunicação síncronas, através de um sistema de mensagens e de conversa (*chat*). Além disso, ele possui uma interface WYSIWIS relaxada, mantendo informações como o posicionamento e o nível de detalhamento (visualização de atributos e métodos) das classes comuns a todos os participantes (quando um altera uma destas propriedades, a modificação é refletida na tela de todos), mas a navegação pelo espaço de trabalho é livre.

Quanto à *workspace awareness*, pode-se afirmar que não há um suporte adequado a esta característica, uma vez que não há como saber onde cada participante está desempenhando suas atividades no momento. Há somente a utilização de cores para a visualização do estado dos elementos bloqueados: as classes que estão sendo editadas aparecem em cores diferentes, de acordo com o participante que está fazendo as modificações, com as cores sendo atribuídas aos participantes por ordem de chegada. Assim, não há uma identificação clara da autoria das atividades. Também não fica clara a metáfora utilizada, assemelhando-se mais à metáfora de um sistema de diagramação.

A divisão das atividades entre os membros no CUTE também não é explícita, nem há a definição de papéis, já que a divisão do processo em tarefas coordenadas e a definição de papéis e responsabilidades é descrita somente na infra-estrutura COPSE,

não estando presente dentro do CUTE. Esta ausência faz com que as ferramentas de comunicação nele presentes sejam obrigatórias para o decorrer do trabalho em equipe, uma vez que será através destas ferramentas que o grupo poderá trocar idéias e organizar suas atividades em conjunto, de forma coordenada. A Fig. 3.9 abaixo mostra uma tela do CUTE, com seus recursos de percepção nela indicados.

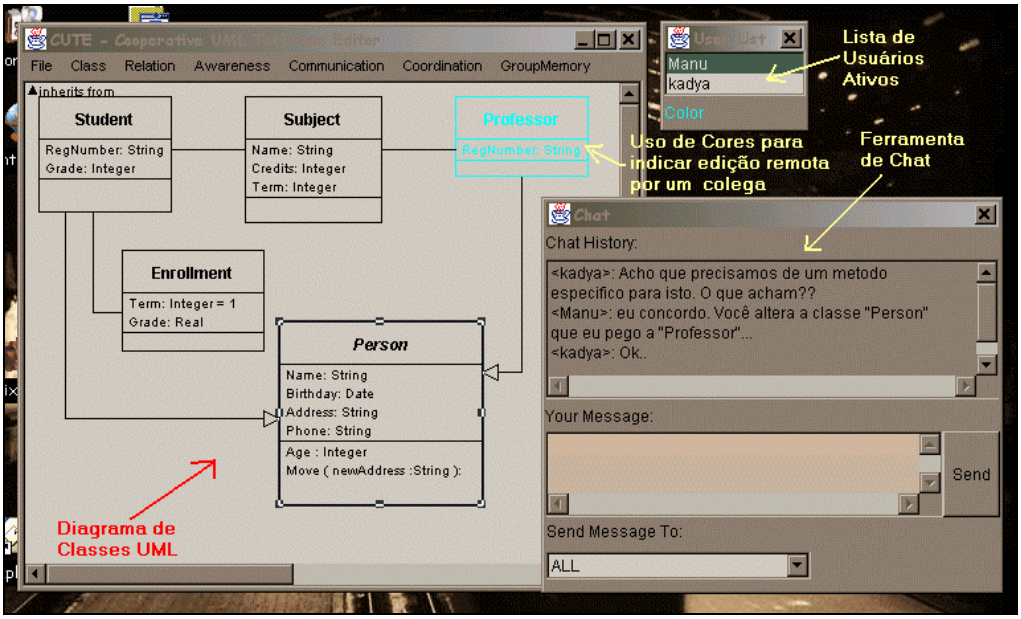


FIGURA 3.9 - Editor CUTE e seus recursos para percepção

3.3.2 PeepHoles / PortHoles / Polyscope (β)

O Polyscope, desenvolvido em 1990, foi o primeiro sistema a explicitamente endereçar o suporte à percepção passiva dentro de um *workgroup*, fornecendo informação de *awareness* independente das tarefas executadas. Seu sucessor, disponível em 1990/1991, o PortHoles, forneceu não somente os benefícios da coordenação da informação de percepção, mas também meios para formar um senso de comunidade em grupos geograficamente distribuídos. Desde então, vários sistemas deste mesmo estilo foram desenvolvidos, como, por exemplo, o Nynex PortHoles [DOU 97]. O Nynex PortHoles é uma extensão para web do Polyscope e do PortHoles e foi criado focando recursos para facilitar o desenvolvimento de redes de trabalho relacionadas, recursos para permitir aos membros serem informados de forma periférica do potencial de colaboração dos colegas, e recursos de comunicação informal, que suportam oportunidades espontâneas e informais para colaboração e interação [LEE 97]. Já o próprio PortHoles suporta a noção de presença, transmitindo periodicamente *snapshots* de vídeo, obtidos por câmeras ligadas às máquinas dos participantes [SMI 98].

O PeepHoles é uma alternativa de baixo custo para fornecer percepção de quem está por perto, para facilitar a interação casual entre os membros. O PeepHoles faz isso através de ícones indicadores da presença e da disponibilidade dos colegas de grupo. A disponibilidade é uma indicação de atividade dos colegas, obtida através do conhecido *daemon* Rusers, não exigindo *hardware* adicional, somente a disponibilidade deste *daemon* na máquina de cada um dos participantes. O PeepHoles também permite aos usuários serem alertados quanto à atividade de um colega, através da notificação pela

emissão de um som quando este se torna ativo, permitindo assim iniciar uma chamada a este colega. Para realizar estas chamadas, o PeepHoles também integra ferramentas de comunicação, como um *link* para o leitor de email [SMI 98].

Estes sistemas têm seu foco totalmente voltado para a questão da presença: quem está conectado no momento e se está ou não ativo, fornecendo também recursos para comunicação através de ferramentas síncronas e assíncronas, com o intuito de criar assim um senso de comunidade e uma base para a comunicação informal dentro do grupo. Todo o funcionamento destes sistemas gira em torno desta noção de presença e da comunicação que pode surgir disso. Em todos eles também é possível perceber características de balanceamento de interface com filtragem, como no PortHoles, através de seus *snapshots* periódicos, e agrupamento, no caso dos ícones indicadores do PeepHoles, que agrupam a noção de presença e atividade dos colegas. Estas duas preocupações chaves fazem deles interessantes exemplos no estudo destas características no suporte à percepção.

3.3.3 WAP (χ)

O objetivo do projeto WAP (*Web Awareness Protocol*) é criar um mecanismo genérico de percepção para uso na Internet. O núcleo deste mecanismo é o desenvolvimento de um protocolo de percepção aberto, como são hoje os protocolos da web. Este novo protocolo será implementado em novos servidores, paralelos aos servidores web (HTTP), não ferindo aos padrões da WWW, mas implicando em diferentes implementações de clientes, que façam uso deste protocolo [ROD 96].

O núcleo do protocolo de *awareness* é gravar a presença dos usuários nas páginas web: quando um cliente requisita um documento para um servidor web, o cliente de *awareness* tenta assinalar (operação de *sign-on*) o servidor de *awareness* co-localizado no site do servidor web, fazendo com que o servidor de *awareness* acrescente aquele cliente à lista de clientes, acessando a URL dada. Quando o cliente web se move para outro documento, o cliente de *awareness* faz o *sign-off* no servidor de *awareness*, o que faz com que este servidor retire o cliente da lista de clientes acessando aquela localização. Com estas operações (*sign-on* e *sign-off*), o servidor de *awareness* pode montar uma base de dados de quais clientes estão acessando que documentos do servidor web co-localizado, e os clientes podem, então, solicitar informações sobre outros clientes assinalados no servidor, através de outras mensagens (quantos e quais clientes estão assinalados para uma URL), embora, em sua primeira versão, o WAP esteja focando só a contagem dos clientes, possibilitando somente saber quão ocupado está o servidor web [ROD 96].

O WAP é outro exemplo de sistema que prioriza a noção de presença dos colegas. Seu objetivo é fornecer a noção de quem está acessando uma dada página web, criando a oportunidade de contato entre seus visitantes. Esta oportunidade pode ser útil entre colegas em uma Intranet e ambientes empresariais distribuídos, mas pode perder sua função fora deste contexto. Além desta, o WAP não fornece nenhuma outra forma de suporte a qualquer das demais questões apontadas neste capítulo, concentrando-se somente na presença. O WAP também não faz nenhuma consideração a respeito da forma de apresentação que os seus clientes podem adotar, mas mantém como vantagem a idéia de um protocolo aberto, que pode futuramente ser estendido para outras questões do suporte à percepção.

3.3.4 POLIAwaC (δ)

O projeto POLITeam tem como objetivo desenvolver um sistema de *groupware* integrado, que forneça um suporte compreensivo para o trabalho cooperativo, especialmente assíncrono e distribuído. Este sistema inclui componentes de *workflow*, processamento coordenado de documentos e tarefas, como edição cooperativa e *workspaces* compartilhados, serviço de notificação e informação, e *gateway* com a WWW [PRO 99]. O POLIAwaC _ *POLITeam Awareness Client* _ é um protótipo que enfatiza o fornecimento de *feedback* compartilhado entre os usuários do sistema POLITeam, fornecendo informações sobre o uso de objetos compartilhados [PRI 99].O POLIAwaC incorpora vários mecanismos para suporte a *awareness*, incluindo realce de ícones de objetos, permitindo percepção periférica da informação de *awareness*, objetos mostrando a localização e a disponibilidade do usuário, e suporte para a apresentação assíncrona de informação [SOH 98].

O POLIAwaC usa a metáfora de *workspace* compartilhado, onde *workspaces* são *containers* de objetos, como documentos, ferramentas e outros *containers*. Estes *workspaces* podem ser públicos ou privados, dependendo do usuário que os acessa, e possuem um proprietário, capaz de convidar novos membros, excluir antigos e passar a propriedade do *workspace* para outro. Como são objetos, os *workspaces* podem ser manipulados (criados, movidos, copiados, etc) e um usuário é considerado estar ativamente trabalhando nestes se o *container* correspondente permanecer aberto no cliente [SOH 98]. Esta metáfora assemelha-se aos *desktops* padrão, usados em diversos sistemas e, segundo Prinz [PRI 99], ela tira proveito das habilidades já existentes dos usuários. Dessa forma, pode-se classificar a metáfora do POLIAwaC como sendo de "sistema", já que o *workspace* compartilhado neste assemelha-se a desktops tradicionais de sistemas com a idéia de pastas que podem conter outros objetos, como documentos e outras pastas, e não se assemelha a idéia de "escritório", não havendo a representação de elementos como sala ou mesa, ou de espaço "físico".

Para a visualização deste *workspace* compartilhado e seu conteúdo, o POLIAwaC fornece três visões especializadas, apresentadas na Fig. 3.10 abaixo: uma hierárquica dos *workspaces*, incluindo os usuários que têm acesso ou que estão ativamente trabalhando, e duas visões mostrando os documentos contidos no *workspace* atualmente selecionado na visão hierárquica, uma visão onde o usuário pode livremente posicionar os ícones, e outra com maiores detalhes sobre os objetos [SOH 98]. Estas 3 visões do *workspace* compartilhado agem em sincronia: ao selecionar um objeto *workspace* na hierárquica, seu conteúdo é mostrado nas demais e quando um objeto é selecionado em uma das outras, a mesma seleção é feita na seguinte. Entretanto, embora existam estas múltiplas visões para cada usuário, o que um está vendo não pode ser visualizado pelos demais, nem informações como os objetos selecionados pelos colegas são acessíveis, o que faz com que estas múltiplas visões do *workspace* compartilhado sejam consideradas também desacopladas, já que a visão que um usuário está tendo é totalmente desacoplada da mantida pelos colegas. Isto não chega a prejudicar o funcionamento deste sistema, já que está em conformidade com seus próprios objetivos, de suporte principalmente ao trabalho assíncrono e não realizar alterações drásticas na interface comumente usada por seus usuários (a metáfora do *desktop*).

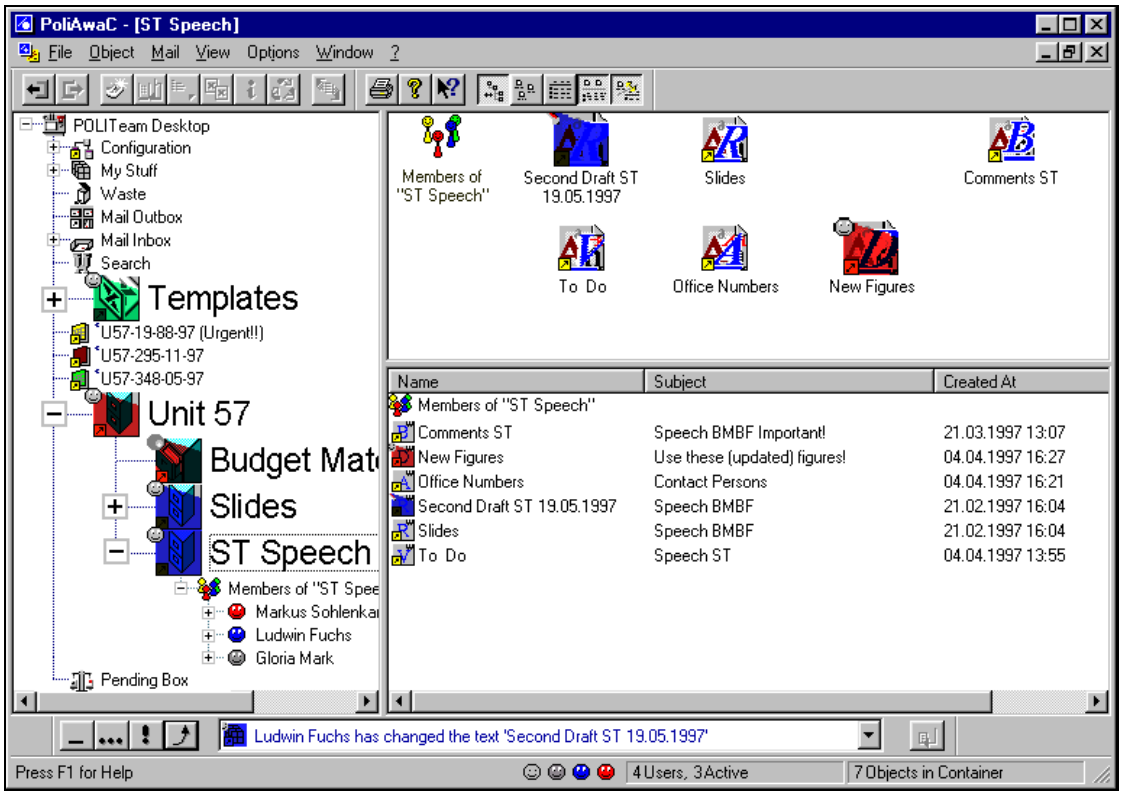


FIGURA 3.10 - As três visões do workspace no POLIAwaC [SOH 98]

O POLIAwaC faz uso dos ícones dos objetos para carregar informação de *awareness*, através de 4 técnicas: a forma (o ícone de um objeto tem sua forma alterada de acordo com seu *status*); a cor (as cores são usadas para representar os usuários, os objetos manipulados por um usuário são sobrepostos por um plano transparente na cor do usuário, e a área coberta por este plano reflete o tempo transcorrido desde o registro do evento); a sobreposição (o tipo de operação realizada em um objeto é indicada por um segundo ícone sobreposto ao original do objeto); e tamanho (os ícones dos objetos modificados são aumentados) [SOH 98], e com o tempo, estas representações são gradualmente reduzidas, de modo a destacar as atividades mais atuais [PRI 99]. Estas representações, que podem ser observadas dentro da Fig. 3.10 acima, podem ser usadas em conjunto ou exclusivamente, de acordo com o nível de intensidade definido para um evento específico. Estes níveis de intensidade de notificação vão desde nenhuma notificação a mostrar o evento em uma caixa de diálogo *pop up*, passando por indicar o *status* do objeto pela cor, mostrar os eventos textuais em uma barra de ferramentas, e alterações e ampliação do ícone do objeto afetado, sendo que os níveis superiores englobam os inferiores [SOH 98].

O POLITeam usa ícones para representar os usuários nas visões do *workspace*. Cada *workspace* não privado possui um pseudo-container “usuários de <nome do *workspace*>”, contendo os ícones dos membros deste *workspace* e sua posição organizacional. Cada ícone é um símbolo genérico legendado com o nome do usuário e desenhado na cor correspondente ao mesmo, com apenas o contorno se este não estiver ativo. O *workspace* raiz na hierarquia é chamado “POLITeam System” e sua seção de usuários mostra a estrutura completa da organização e todos os usuários potenciais e ativos [SOH 98].

Assim, uma certa noção de presença é fornecida pelo sistema, mais no sentido de oferecer uma oportunidade aos membros. Entretanto, quanto à comunicação, não há um suporte adequado, limitando-se principalmente ao email (assíncrona), e como há a apresentação de algumas informações organizacionais e a figura do proprietário do *workspace*, pode-se afirmar que existe uma certa preocupação com a questão dos papéis, embora ainda limitada a somente estas poucas informações.

O POLIAwaC fornece três elementos básicos de interface para fornecer apresentação assíncrona da informação de *awareness*: uma barra de eventos, uma caixa de diálogo de eventos e uma caixa de diálogo de histórico. A barra de eventos fornece tanto o histórico quanto a notificação quase imediata de eventos e a possibilidade de gerar informações de *awareness* relacionada a um objeto. Esta barra, apresentada na Fig. 3.11 abaixo, consiste de uma barra de texto no estilo *drop down*, que mostra no topo o evento mais recente e, quando aberta, mostra uma lista com os últimos eventos em ordem cronológica, além de botões, que fornecem um *link* direto entre os eventos e os objetos relacionados. Cada evento mostrado na barra de eventos é mostrado na cor associada ao usuário que o gerou, com uma descrição textual do evento e o ícone do objeto associado. Os usuários também podem entrar com uma frase no campo de texto da barra de eventos, a qual será distribuída como um evento associado com o objeto selecionado. Isto permite fornecer informação de *awareness* que não pode ser coletada automaticamente pelo sistema. Outra vantagem da barra de eventos é que ela também pode ser usada como uma janela *stand-alone* sobre as janelas de outras aplicações, para o monitoramento das atividades cooperativas [SOH 98].

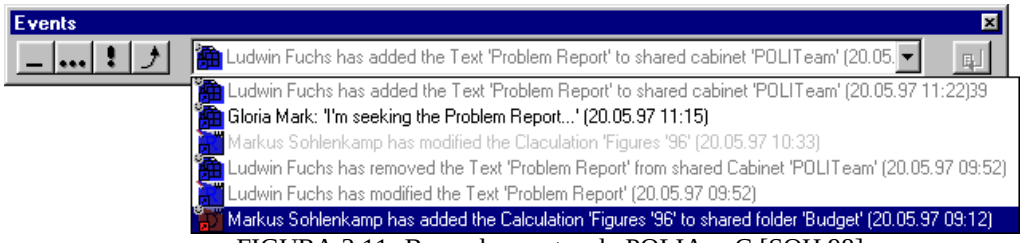


FIGURA 3.11- Barra de eventos do POLIAwaC [SOH 98]

Já a caixa de diálogo de eventos usa os mesmos mecanismos da barra de eventos, exceto a lista de histórico que está sempre expandida, e é usada para mostrar eventos com alta prioridade, sendo lançada automaticamente, ficando em primeiro plano para estes eventos. A janela de histórico é uma aplicação independente do POLIAwaC que busca do servidor o histórico de um objeto de forma textual e pode ser invocada ativamente pelo usuário. A descrição de cada evento é colocada na cor do autor, atributos adicionais podem ser selecionados e todos os eventos de *workspace* privados são resumidos em uma única entrada anônima [SOH 98].

Através destes recursos, os usuários do ambiente do projeto POLITeam têm acesso a eventos que ocorreram tanto no passado quanto no passado contínuo, este último reforçado pelo uso de cores, ícones e alterações nos ícones, descritas anteriormente. Assim, há junto ao POLIAwaC uma apresentação posterior dos eventos, e a conseqüente persistência alta dos dados, que são armazenados junto ao servidor. Também se pode observar que toda a informação de *awareness* encontra-se atrelada aos objetos compartilhados e às ações sobre estes objetos, as quais geram os eventos relacionados.

A filtragem da informação de *awareness* no POLIAwaC é feita por "*awareness* profiles", baseados na idéia de "situações de trabalho". Qualquer objeto define um número de atividades e uma variedade de situações de trabalho como "trabalhando no documento" ou "acessando o *container* pai", nas quais pode estar envolvido. Os detalhes de preferência de *awareness* são definidos usando então os "*awareness profiles*", que podem ser anexados a objetos simples, coleções ou classes inteiras de objetos. O sistema somente informará sobre os eventos que estiverem de acordo com o conjunto de *profiles* assinados pelo usuário, o qual também pode especificar a intensidade de apresentação para os eventos. Estes *profiles* são objetos compartilhados, logo, os usuários podem tanto criar novos quanto assinar os já existentes. Através deles é possível a diversificação individual do *feedback* do sistema e também a inscrição de um conjunto de *profiles* em um grupo, estabelecendo uma forma recíproca de *awareness* para o grupo [SOH 98], [PRI 99].

Além desta filtragem feita através dos "*awareness profiles"*, pode-se considerar que o POLIAwaC faz ainda um agrupamento de informações em sua interface, através dos ícones, que por suas mudanças na forma, tamanho, cor e sobreposição de outros ícones, informa perifericamente o usuário sobre o tipo de alterações e operações feitas nos documentos, seu autor e o tempo transcorrido.

Para criar e editar os documentos todo o ambiente POLITeam faz uso de aplicações padrões, que são integradas ao sistema. Estes documentos são armazenados no servidor central e recuperados quando os usuários os abre. A edição propriamente dita é feita em uma cópia local com a aplicação padrão, enquanto a cópia no servidor fica bloqueada para edição dos demais usuários. Entretanto, estas aplicações utilizadas são projetadas para uso mono-usuário e não cooperativo. Por este motivo, a barra de eventos foi projetada para também poder funcionar como uma janela *stand-alone*, que pode ser instruída a ficar sobre as demais janelas, deixando sempre visível o evento mais recente de interesse do usuário. A visão hierárquica também pode ter seu tamanho reduzido, permitindo a visualização da hierarquia de *workspaces* completa, mesmo sob as aplicações [SOH 98].

O uso de aplicações padrão para edição dos documentos afeta diretamente o nível das informações de percepção das atividades. Estas atividades são apenas visíveis em um macro-nível, uma vez que somente é possível saber que foram feitas determinadas operações nos objetos como um todo, como editar, apagar ou mover um documento inteiro, e não é possível saber, por exemplo, que parte do documento foi alterada, mas apenas que ele foi alterado.

Convém mencionar aqui também que o projeto POLITeam tem como plataforma base o LinkWorks, um *groupware* cliente/servidor e orientado a objetos produzido pela Digital. Entre suas vantagens está permitir a integração de qualquer aplicação rodando nas plataformas que utiliza, a configuração de suas funcionalidades, via interface gráfica e a inclusão de novas funcionalidades via interface APO _ *Appliaction Plus Objects*, que permite o acesso a suas estruturas de dados internas. Especificamente quanto à percepção, o LinkWorks possui apenas um suporte rudimentar nativo, permitindo o registro dos usuários sobre os objetos de seu interesse e a apresentação dos eventos que ocorrem sobre estes objetos apenas por meio de uma caixa de diálogo com a descrição textual do evento, além de um "log" de eventos global, que pode ser aberto assincronamente pelo usuário, e uma caixa de diálogo simples com os usuários logados [PRI 99], [SOH 98].

3.3.5 DIVA (ε)

O DIVA é um protótipo de ambiente de escritório virtual, com foco no fornecimento integrado de suporte para percepção dos colegas, tanto operando em modo de trabalho síncrono quanto assíncrono. Seu principal objetivo é realizar um projeto de interface com o usuário consistente, que integre diferentes aspectos do trabalho cooperativo e permita transições fáceis entre diferentes modos de trabalho [SOH 98].

O DIVA usa uma metáfora de escritório. Ele modela quatro elementos principais: "pessoas", "salas", "mesas" e "documentos", além de elementos adicionais, como "notas" e "lata de lixo". As "pessoas" representam os usuários do sistema e são implementados por pequenos *snapshots* com o nome do usuário. As "salas" são *containers* para pessoas, mesas e documentos, podendo estar abertas, fechadas ou trancadas, fornecendo assim diferentes níveis de acesso e visibilidade. Nas salas também é feito o controle da comunicação de áudio e vídeo entre os colegas, de modo que as pessoas em uma mesma sala virtual DIVA podem ver e ouvir umas as outras, e sempre que alguém entra em uma destas salas, são estabelecidas ligações de áudio e vídeo entre o recém-chegado e os seus ocupantes anteriores, de forma análoga ao que ocorre em uma sala real [SOH 98].

As "mesas" funcionam como local para o trabalho dentro das salas, controlando também o modo de acoplamento da cooperação, além de poder ser usada para preservar o contexto de trabalho [SOH 98]. Várias pessoas podem trabalhar em uma mesa sobre os mesmos documentos ou podem trabalhar nestes documentos em mesas (e salas) distintas, o que define diferentes modos de acoplamento entre as interfaces dos usuários, o que pode ser caracterizado como a presença de múltiplas visões.

O DIVA fornece tanto suporte ao trabalho cooperativo síncrono quanto assíncrono, oferecendo ferramentas para tarefas compartilhadas, facilidades para suporte à comunicação e mecanismos de interface para o fornecimento de percepção dos colegas. Durante o modo de trabalho síncrono, a janela do escritório virtual (que contém todas as salas do ambiente) fornece um *overview* das atividades dos colegas por todo o escritório virtual, enquanto a janela da sala virtual fornece informações mais detalhadas sobre a sala em particular. No escritório virtual, os ícones dos usuários são apresentados dentro da sala virtual onde estão, dando a impressão que os usuários estão "realmente" na sala. Com isso, é possível observar a localização dos colegas, quem está falando com quem e sua disponibilidade para contato, e se a maioria das salas estiver aberta, uma rápida olhada pelo escritório virtual fornece um *overview* das atividades realizadas. Além disso, os usuários são capazes de controlar não só as informações sobre si mesmos, distribuídas para os colegas, quanto o seu nível de disponibilidade para comunicação [SOH 98].

Estas características dão uma noção sobre a presença e a disponibilidade dos colegas, as quais, juntamente com as ferramentas de comunicação, via canais de áudio e vídeo, durante o modo de cooperação síncrono (ferramentas síncronas), e via notas que podem ser deixadas para os colegas, no modo de trabalho assíncrono (ferramentas assíncronas), funcionam tanto como uma obrigatoriedade, no primeiro caso, quanto uma oportunidade, na última situação. O uso da metáfora de escritório, com salas e mesas, onde ficam dispostas as pessoas, documentos, etc., também dá uma noção de *workspace awarenes*, limitada a informações como em que sala e mesa uma pessoa está e os movimentos feitos pelas pessoas dentro de uma sala.

Além da indicação de presença dos colegas nas salas, estas também são marcadas com uma espécie de barra vermelha em seu topo quando nelas encontram-se documentos os quais os usuários tenham acesso. Uma vez dentro de uma sala virtual, os usuários podem ver quem está trabalhando com quem, através da indicação das mesas, e em que documentos, além de animações para indicar o movimento dos usuários na sala e um sinal sonoro, indicando o ingresso de novos membros. Os documentos também carregam informação visual de percepção. Seus ícones indicam visualmente o *status* do documento, por exemplo, documentos não abertos pelo usuário, mas que outros estão editando, mesmo que em uma sala diferente, são marcados como em uso e apresentados com um fundo vermelho. Adicionalmente, cada pessoa possui ainda cores de assinatura, que as identificam durante o trabalho fortemente acoplado [SOH 98]. Assim, pode-se perceber algumas características de agrupamento de informações de percepção presentes na interface do DIVA, a qual é apresentada na Fig. 3.12 abaixo.

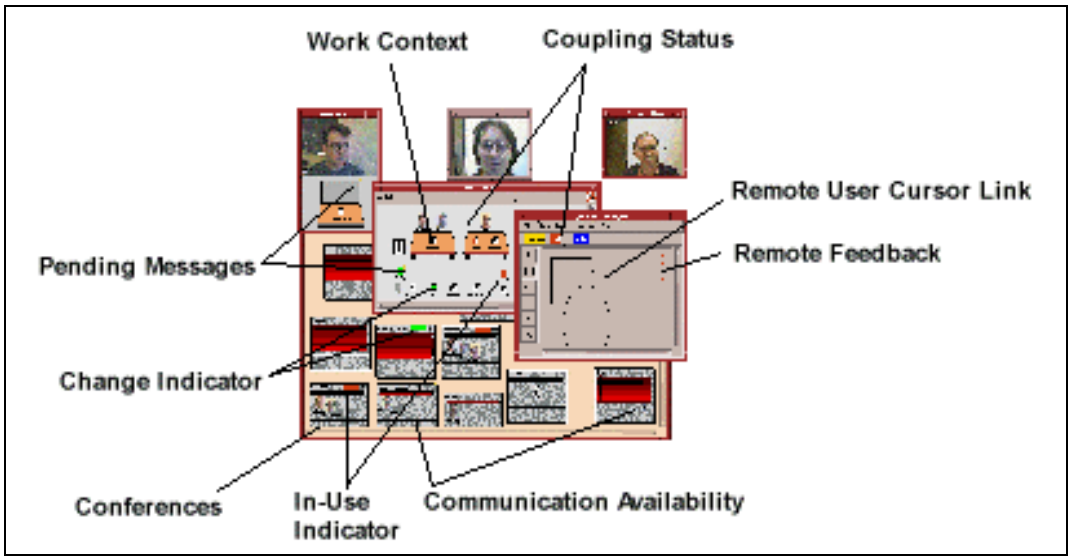


FIGURA 3.12 - Workspace DIVA [SOH 98]

Além disso, existem junto ao DIVA ferramentas para manipulação dos documentos, que permitem criá-los, editá-los, etc., para a edição síncrona, e o revezamento coordenado do trabalho e operações como *merge* de versões divergentes permitem o trabalho assíncrono sobre os mesmos documentos. Estas ferramentas permitem aos usuários saberem o que seus colegas estão fazendo, principalmente durante uma sessão de edição compartilhada, e determinar quando e como os objetos compartilhados foram alterados [SOH 98]. Isto fornece tanto uma noção em macro-nível das atividades, quando em modo assíncrono, quanto em micro-nível, em modo síncrono, além de fornecer informações sobre eventos em um passado contínuo (informações ainda válidas) e presentes, de acordo com o modo de trabalho utilizado e, como conseqüência, apresenta a persistência destas informações por um certo período de tempo.

3.3.6 *Alliance /Byzance* (ϕ)

O Alliance é um editor cooperativo assíncrono que permite a um grupo de pessoas, geograficamente distribuídas e conectadas a Internet, editarem conjuntamente documentos estruturados. Seu funcionamento baseia-se na fragmentação do documento em partes, de acordo com a atribuição de regras de edição (papéis) para os autores, definindo seus direitos sobre cada um dos fragmentos. Esta fragmentação é feita com auxílio da API Thot, uma API para o desenvolvimento de aplicações que manipulem documentos estruturados. O Thot fornece um conjunto de funções de manipulação da estrutura lógica e do conteúdo destes documentos, e permite ainda às aplicações incluírem funções extras pelo mecanismo de "*call backs*", além de serviços adicionais para interface gráfica [DEC 98a], [DEC 98b], [QUI 99a].

Um documento no Thot é representado por sua estrutura lógica, chamada de "Árvore Abstrata", uma estrutura essencialmente hierárquica, acrescida de relações suplementares, formando também um hipertexto. Esta estrutura segue algumas regras, uma espécie de tipo do documento, que ajudam (ou obrigam) os usuários a produzirem documentos conforme este seu tipo. Fazendo uso desta estrutura do Thot, o Alliance divide seus documentos em fragmentos, os quais são dispostos nos sites de cada um dos participantes. Esta distribuição segue a divisão do documento entre os participantes, de acordo com seus papéis, de modo que somente um participante por vez possa ser capaz de alterar um fragmento. Isto é obtido através de um esquema de cópias *master* e *slaves*, a chamada "regra da unicidade da cópia *master*": para cada fragmento existem várias cópias ditas *slaves*, uma em cada um dos sites dos participantes, e uma cópia dita *master*, a única que pode ser alterada, colocada somente no site do participante que tiver o poder de escrita. Este poder é dado pelos papéis de "gerente", capaz de ler e modificar o documento e ainda determinar os papéis de cada co-autor, e "escritor", capaz de ler e alterar o fragmento. Os demais papéis são "leitor", com acesso somente para leitura, e "nulo", que não possui acesso algum sobre o fragmento, nem leitura nem escrita [DEC 98b], [SAL 98].

Ainda com relação aos papéis, cada co-autor tem um conjunto de papéis ditos potenciais para cada fragmento do documento, que são aqueles papéis definidos pelo gerente e representam todos os possíveis papéis que o co-autor poderá desempenhar, além de um conjunto de papéis ditos efetivos, que são os papéis que estão realmente sendo desempenhados pelos participantes. À medida que o trabalho vai sendo feito, os participantes podem ir trocando seus papéis efetivos, de acordo com os seus papéis potenciais, e isto poderá causar modificações na fragmentação do documento, como a divisão ou *merge* de fragmentos, a fim de adaptar o documento como um todo à regra da unicidade da cópia *master*. Graças a este mecanismo, é possível a vários co-autores dentro do Alliance editarem fragmentos diferentes do mesmo documento em um mesmo intervalo de tempo [SAL 98].

Dentro do Alliance, o suporte à percepção, ou consciência de grupo, como é chamada por Salcedo [SAL 98], permite controlar e atuar sobre o ambiente de edição para que os co-autores possam ser advertidos de modificações nos fragmentos, através de um sistema de notificação. Este sistema permite a cada co-autor regular as notificações de modificações de seus colegas sobre os fragmentos e foi concebido para facilitar a integração com o processo de produção do documento, simplificar sua manipulação pelos co-autores, refletir o estado da cooperação em um dado momento, e ser eficaz na atualização dos fragmentos [SAL 98]. Este sistema permite que os co-autores controlem seus próprios níveis de notificação, optando entre a consulta com

atualização automática, aberta ou travada. No primeiro caso, as modificações feitas por um colega no fragmento são refletidas na interface do usuário tão logo estejam disponíveis, o que não eqüivale à edição síncrona, pois pode haver um atraso entre os colegas realizarem as modificações e as divulgarem para os demais. No caso da consulta aberta, o usuário só poderá ver as alterações no fragmento após requisitá-las voluntariamente, e na consulta travada, o usuário opta por não ser advertido de alterações no fragmento, nem quer levá-las em conta durante o seu trabalho.

Estas opções do usuário quanto ao sistema de notificação, bem como os papéis efetivos desempenhados por ele em cada fragmento, são indicados graficamente por ícones integrados à janela do editor no início de cada fragmento. As possíveis alterações, tanto nas opções do sistema de notificação, quanto alterações nos papéis e nos próprios fragmentos (criação de novos, *merge*, etc), são representadas por ícones diferentes, cuja transição entre um e outro ícone é definida por um autômato finito predeterminado [SAL 98].

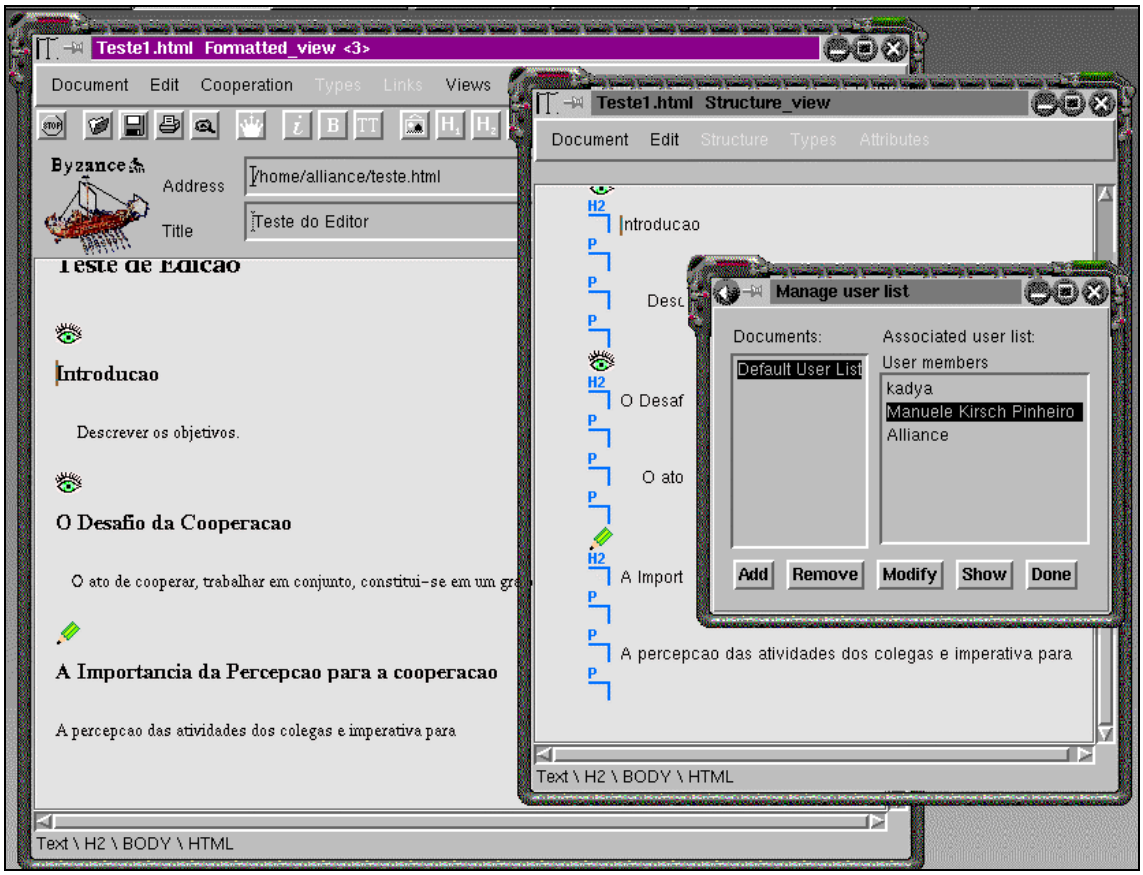


FIGURA 3.13 - Editor Cooperativo Byzance

O mecanismo de suporte à percepção implementado dentro do Alliance através da notificação das modificações pode ser caracterizado por apresentar as atividades em um macro-nível, uma vez que não é possível ver exatamente quais alterações foram feitas pelos colegas nos fragmentos, mas somente ver o novo fragmento inteiro. Ele também apresenta eventos em um passado contínuo, pois somente eventos que ocorreram em um dado momento no passado e continuam válidos são apresentados, posteriormente a sua ocorrência (apresentação posterior), o que representa uma certa persistência nas

informações. Esta persistência é limitada à manutenção das cópias *slaves* dos fragmentos, à atualização destas por decorrência de modificações nas respectivas cópias *master* e pela manutenção das opções de notificação por fragmento.

Além disso, o Alliance apresenta uma metáfora de sistema, assemelhando-se a um editor de textos tradicional, e também apresenta em sua interface o recurso de múltiplas visões, permitindo a visualização e a edição do documento em várias visões. O Alliance também possui uma diferenciação nas informações que apresenta de acordo com o papel desempenhado pelo participante, especialmente no tocante ao papel "nulo".

O Alliance foi utilizado recentemente para a construção de um novo editor cooperativo, o Byzance, apresentado na Fig. 3.13. O Byzance é um editor HTML cooperativo, permitindo a edição cooperativa de páginas web de forma semelhante ao Alliance. O Byzance possui as mesmas características do Alliance, essencialmente quanto à percepção. Maiores informações quanto ao Byzance podem ser encontradas junto a páginas do grupo Opera, responsável por seu desenvolvimento [OPE 99].

3.3.7 *SISCO-Rio* (γ)

O SISCO é um sistema distribuído e assíncrono para o apoio à preparação de reuniões. Seu objetivo é tornar as reuniões face a face mais eficientes, através de um ambiente de pré-reunião, que permite a discussão dos temas a serem tratados na reunião, sem a obrigatoriedade da tomada de decisão desta. Seu objetivo não é o de substituir a reunião, mas apenas torná-la em eficiente pela discussão de seus temas antes que ela realmente ocorra, já que reuniões face a face normalmente têm associados altos custos em sua produção, o que sugere que estas deveriam se tornar tão efetivas quanto possível, e uma boa preparação deixaria os participantes mais atentos aos objetivos e ao curso esperado para a reunião [BEL 95].

Para suportar a discussões durante uma pré-reunião, o modelo SISCO original assume que as pessoas envolvidas nesta fase de pré-reunião estão distribuídas e podem trabalhar assincronamente, embora a comunicação com os outros participantes ainda seja fundamental, pois estas pessoas precisam contribuir dentro de um contexto de idéias e devem ter as contribuições anteriores disponíveis [BEL 95]. Para tanto, o modelo SISCO adota uma extensão do modelo de argumentação IBIS, formado pelos elementos "questão", "posição" e "argumento", e os relacionamentos entre eles, acrescido de elementos para a coordenação, agenda do grupo e elementos para apoiar contribuições que não se encaixam nas categorias previstas pelo IBIS. Sua arquitetura é composta por uma memória de grupo que armazena as contribuições feitas durante a pré-reunião, um mecanismo de comunicação e uma interface com a memória de grupo. O SISCO-Rio é a implementação de um subconjunto deste modelo original SISCO com algumas limitações, como a ausência de alguns objetos e a presença somente das funcionalidades de coordenador e contribuinte. O SISCO-Rio faz a manutenção da memória do grupo com auxílio do SGBD CA-Ingres e incluiu uma interface Web com o usuário, também incluindo o mecanismo de comunicação através de um sistema de mensagens, o qual permite aos usuários trocarem mensagens particulares, além da comunicação feita através da própria memória de grupo. Através do servidor Web, o SISCO-Rio resgata as informações da memória de grupo armazenadas no SGBD, as quais são apresentadas para os usuários pela interface Web [CAV 97].

Cada pré-reunião dentro do SISCO-Rio passa por três momentos: primeiro a criação desta pelo coordenador, seguido do registro dos participantes e, por fim, define-se a agenda que guiará as discussões através, principalmente, de suas prioridades e itens, com prazos e objetivos bem definidos. Além do papel de coordenador, que cria e configura as pré-reuniões, o SISCO-Rio define também o papel de contribuinte, que são os participantes que podem recuperar e adicionar elementos à discussão, como uma forma de se prepararem para a reunião subseqüente. Estes contribuintes iniciam sua participação acessando os elementos já existentes e a partir daí poderão inserir novos elementos à memória de grupo. Através da contribuição destes elementos (questões, posições, argumentos e notas), eles definem e discutem suas idéias, seguindo uma estrutura hierárquica preestabelecida, onde o nível mais alto contém as pré-reuniões das quais participa o usuário, seguidas pelos seus itens, objetivos e depois elementos correspondentes [CAV 97].

Os mecanismos de suporte à percepção dentro do SISCO-Rio agem no sentido de informar aos participantes sobre novos elementos quando este acessa o sistema, no momento em que os elementos da memória de grupo são recuperados e apresentados para o usuário, via interface *web*. São considerados três possíveis estados para cada elemento: “novo”, para elementos novos, não vistos pelo usuário, “conhecido”, para elementos de cuja existência o usuário já tem conhecimento, mas que ainda não viu o seu conteúdo, e “velho”, para elementos cujo o conteúdo já é de conhecimento do usuário. Para os dois primeiros, há ícones associados (o símbolo “new” para elementos novos e um bloco de notas para os conhecidos), que permitem ao usuário reconhecer rapidamente os elementos que ainda não foram lidos [CAV 97]. A Fig. 3.14 abaixo marca um exemplo do uso do SISCO-Rio, com várias pré-reuniões (PM _ *Pre-Meeting*) em curso de forma assíncrona. A primeira encontra-se expandida, com um item, seu objetivo e várias contribuições visíveis.

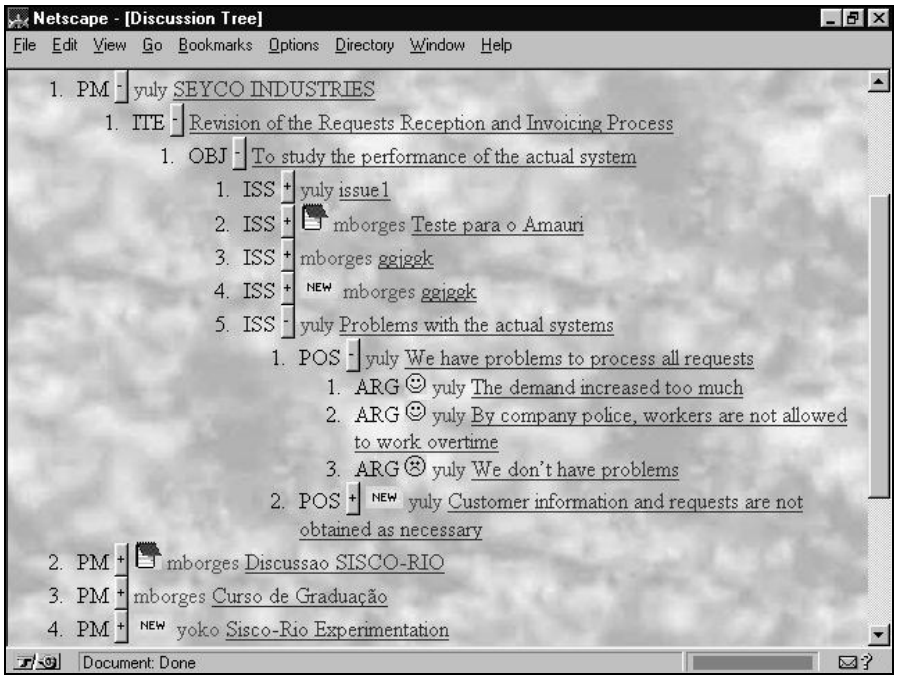


FIGURA 3.14-Exemplo de percepção do conteúdo da memória de grupo pelo SISCO [BOR 99b]

Este mecanismo de notificação implementado no SISCO-Rio apresenta as informações sobre as atividades dos participantes em um macro-nível, já que somente são apresentados os elementos inteiros, com a indicação diferenciada para "novo" ou "conhecido". Além disso, o SISCO-Rio trabalha com a noção de papéis, diferenciando o coordenador dos contribuintes, embora esta diferenciação se limite somente aos poderes atribuídos ao primeiro e não ao mecanismo de notificação diretamente.

O uso de memória de grupo, da forma como é feito dentro do SISCO-Rio, age no sentido de facilitar o fornecimento de percepção de eventos ocorridos no passado ou no passado contínuo, com apresentação posterior (no momento em que o usuário ingressa no sistema), possuindo uma persistência alta, implementada através do uso do SGBD para a manutenção da memória de grupo.

Quanto à comunicação entre os participantes, o SISCO trata a presença dos colegas no sistema como uma oportunidade, apresentando somente uma ferramenta de comunicação síncrona, por troca de mensagens rápidas e não armazenadas entre os participantes. Ele também considera a própria memória de grupo como uma forma de comunicação assíncrona entre os membros do grupo. Já quanto à interface com o usuário, o SISCO-Rio adota uma interface web totalmente desacoplada da interface visualizada pelos demais colegas.

3.3.8 GroupKit (η)

O GroupKit é um *toolkit* para a construção de *groupware* síncronos e distribuídos para a realização de conferências entre duas ou mais pessoas. Ele suporta conferências multi-ponto, distribuídas, em tempo real, baseadas em texto ou gráficos (o GroupKit não suporta áudio e videoconferências). Inclui também aplicações de conferência propriamente ditas, que são as ferramentas de *groupware* atuais, e gerentes de sessão, que permitem aos usuários criarem e gerenciarem suas conferências [ROS 96],[ROS 97].

O GroupKit é formado por quatro elementos chaves: uma infra-estrutura de *Run-Time*, que automaticamente gerencia a criação e comunicação dos processos distribuídos que compõem as sessões de conferência; um conjunto de abstrações de programação, que permite controlar o comportamento destes processos, permitindo aos mesmos tomarem ações nas mudanças de estado e compartilharem dados relevantes; um conjunto de *widgets*, elementos de interface que permitem a adição de importantes características de interface a um *groupware*; e alguns gerentes de sessão, que permitem as pessoas gerenciarem suas reuniões, de forma desacoplada das aplicações de *groupware*, e que também podem ser construídas pelos desenvolvedores para se adaptarem ao estilo de trabalho do grupo [ROS 96]. As próprias aplicações de conferência disponíveis no GroupKit também são desacopladas dos gerentes de sessão fornecidos. Além disso, os desenvolvedores não precisam se incomodar, ao criar suas aplicações de *groupware* com o GroupKit, com a criação de novos processos ou com a gerência da comunicação entre os existentes, mas apenas precisam ter conhecimento que estão trabalhando com processos que serão replicados em várias máquinas, um para cada participante da conferência, para com este conhecimento poder utilizar os recursos do GroupKit que forem mais convenientes e deixar que a infra-estrutura de *run-time* faça o resto. A Fig. 3.15 abaixo, mostra o conjunto de aplicações de conferência que fazem parte da distribuição padrão do Groupkit [ROS 97], [GRE 00].

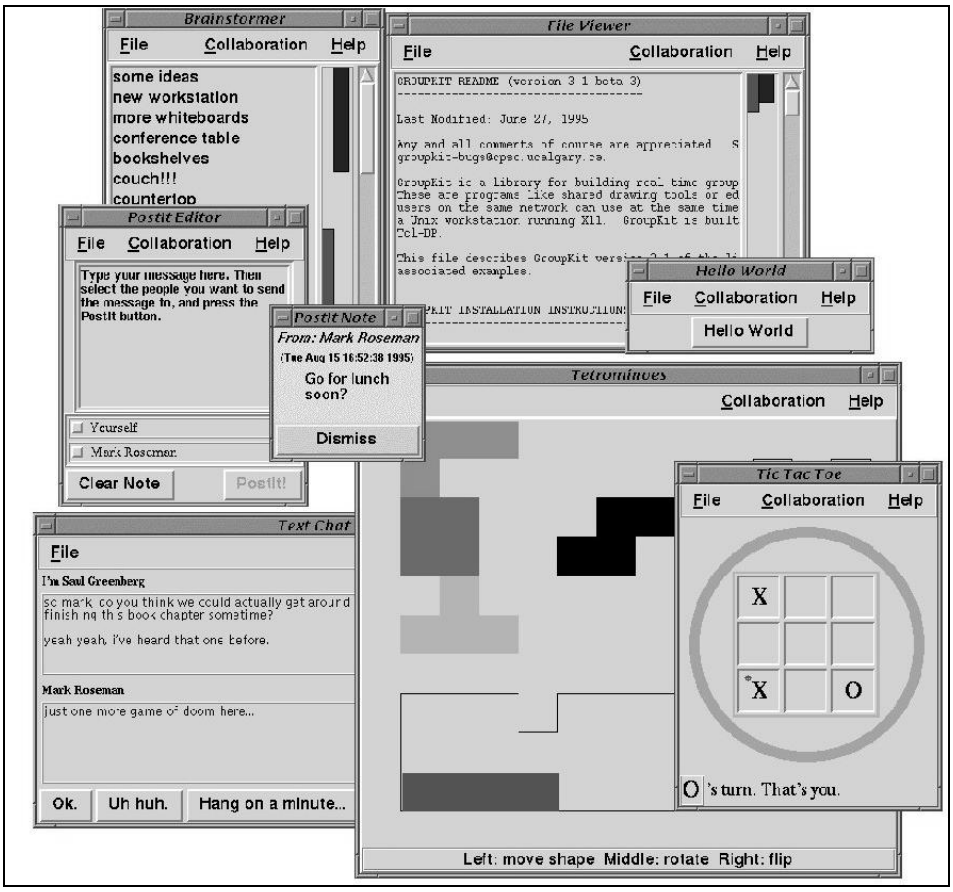


FIGURA 3.15 - Ferramentas da distribuição [ROS 97]

A infra-estrutura de *run-time* controla os múltiplos processos distribuídos que formam as sessões de conferência. Ela é responsável por criar, interligar e localizar os processos, por fazer a gerência de comunicação, com a manipulação de *sockets* e *multicast*, além da gerência das próprias sessões de conferência. Ela se divide em três tipos de processos distribuídos: um único processo centralizado, chamado "Registar", um típico *daemon* Unix, responsável por manter as conferências que estão acontecendo e quem está presente em cada uma; os gerentes de sessão, processos replicados que permitem aos usuários manipularem (criarem, apagarem, monitorar, ingressar ou deixar) as conferências, fornecendo não só uma interface para isso, mas também uma política para tal manipulação; e as aplicações de conferências, que são os programas GroupKit invocados pelos gerentes de sessão quando os usuários ingressam em uma conferência. Junto a cada usuário roda um gerente de sessão, que se conecta ao Registar para saber as conferências disponíveis e seus participantes. À medida que os usuários vão ingressando nas conferências, através de seus gerentes de sessão, novos processos das aplicações vão sendo disparados e as conexões de rede entre estes vão sendo feitas de modo que todo processo em uma conferência terá uma conexão com todos os demais processos da mesma conferência [ROS 96], [ROS 97].

Com esta infra-estrutura mantida pelo GroupKit, o código das aplicações de conferência pode ser construído com o mínimo de ações para a criação e comunicação dos processos, podendo se limitar somente ao uso das abstrações de programação. O GroupKit fornece três tipos de abstrações de programação: os *Multicast Remote Procedure Calls* (Multicast RPCs), eventos e a estrutura Environments. O primeiro

consiste em uma extensão do mecanismo de RPC, que permite enviar comandos de um processo local para outro remoto. O GroupKit fornece várias chamadas deste tipo, possibilitando ao desenvolvedor executar um comando ou procedimento em todos os processos participantes da conferência, incluindo ou não o processo local, ou em um processo determinado, sem para isso precisar especificar o endereço exato da máquina e processo destino.

Os eventos fornecem um meio para as aplicações de conferência serem notificadas quando certas mudanças acontecem. Cada evento consiste de um tipo e um conjunto de pares atributo/valor, os quais fornecem as informações sobre o evento. Com isso, os programadores podem capturar tipos particulares de eventos e manipulá-los, especificando-se para isso um código de *callback* [ROS 96]. A própria infra-estrutura GroupKit é responsável por gerar três tipos de eventos, um indicando o ingresso de novos participantes na conferência (newUserArrived), outro quando um usuário deixa a conferência (userDelete) e um terceiro para a atualização de retardatários em uma conferência já em progresso (updateEntrant). Este último é um evento especial, enviado pelo GroupKit a somente um dos processos participantes, escolhido arbitrariamente, quando um novo processo daquela conferência é iniciado. O processo que recebe este evento deve responder ao retardatário enviando toda informação necessária para que este último fique atualizado. Este evento também é utilizado para tornar uma conferência persistente. Quando o último membro de uma conferência sai, ele é questionado se deseja apagar a conferência ou salvar seu conteúdo. Caso ele opte por salvar, o GroupKit envia o evento updateEntrant para esta última instância da conferência, a qual deverá responder para um servidor de persistência especial com as mensagens de atualização da conferência. Quando o próximo usuário entrar na conferência, estas mesmas mensagens armazenadas serão repassadas para o novo usuário pelo servidor de persistência, na mesma ordem em que ele as recebeu do último usuário [ROS 97].

A estrutura Environments é uma estrutura de dados no estilo dicionário, contendo chaves e valores associados. As instâncias de Environments, rodando nos diferentes processos de conferência, podem se comunicar entre si, comportando-se como uma estrutura de dados compartilhada e permitindo às aplicações GroupKit se comunicarem através delas. A estrutura Environments pode ser usada para armazenar e recuperar informações, organizadas em forma de uma árvore de chaves relacionadas. Ela também permite aos programadores ligarem códigos de *callback*, de modo a permitir uma notificação quando uma nova informação é colocada, ou quando a estrutura é alterada ou removida. Assim, as alterações no ambiente de um processo da conferência podem ser propagadas para os outros processos na mesma sessão da conferência [ROS 96]. O GroupKit mantém um certo número de Environments internamente. Um deles é o chamado “users”, que mantém as informações sobre os usuários na conferência, tanto o local quanto os remotos. Estas informações incluem não só o nome, e *host* de cada usuário, como também uma cor determinada para o usuário e um número único que o identifica dentro da conferência [ROS 97].

Quanto aos *widgets*, o GroupKit fornece uma coleção de *widget* para ferramentas de *groupware*. Essa coleção inclui elementos para fornecer informações sobre os colegas, como o *widgets* “Participantes”, o qual lista os participantes de uma conferência de forma dinâmica, sendo atualizado sempre que um membro entra ou sai desta, fornecendo também informações sobre um participante selecionado pelo usuário (este *widget* foi apresentado anteriormente na Fig. 3.5). O GroupKit também inclui em seus *widgets* “Telepointers” (também conhecidos como múltiplos cursores), como

mecanismo de comunicação de gestos. Segundo Roseman e Greenberg [ROS 96], os gestos são um mecanismo de comunicação rico, que permite aos participantes indicar relações entre artefatos, chamar a atenção para um artefato em especial, mostrar suas intenções sobre o que está para fazer, etc., e os Telepointers constituem uma forma simples e efetiva de comunicar estes gestos. Através deles é possível visualizar a movimentação do mouse feita pelos colegas, o que fornece uma boa idéia do que está sendo feito por cada um. Dentro do GrouptKit, estes Telepointers podem ser criados e anexados a outros *widgets*, onde são apresentados pequenos cursores com a posição relativa de cada um dos demais participantes da conferência.

O GroupKit também fornece o *widget* de "Multi-Users Scrollbar", que consiste em uma barra de rolagem convencional e um conjunto de barras mostrando a posição relativa dos colegas dentro do documento. Estas barras acompanham o movimento dos demais participantes, alterando sua posição e tamanho à medida que os usuários vão alterando sua posição no documento ou tamanho de janela. Através destas barras é possível saber onde os colegas estão trabalhando e facilmente alinhar-se aos mesmos [ROS 97]. Além das multi-users scrollbars, também há um outro *widget*, chamado "Gestalt Viewer", que permite a localização dos colegas em um grande documento. O Gestalt Viewer, apresentado anteriormente na Fig. 3.4, consiste em uma miniatura do documento sobreposta por caixas coloridas, mostrando o ponto de vista atual de cada participante na sessão. Esta miniatura fornece a localização dos demais participantes e uma visão estrutural do documento, o que pode ajudar o usuário a melhor entender onde e em que seus colegas estão trabalhando [ROS 96].

Além destes *widgets*, o GroupKit fornece ainda outros, como uma barra de menus padrão, com acesso ao *widget* Participantes, e um *class builder* simples, que permite aos desenvolvedores a construção de novos *widgets*.

Através destes *widgets*, o GroupKit permite que as aplicações de conferências construídas sobre suas bases possam apresentar informações de percepção. Através, por exemplo, dos telepointers, é possível apresentar informações sobre as atividades sendo realizadas em um micro-nível, mostrando em detalhes o que os colegas estão fazendo no momento. Por ser um toolkit projetado para o desenvolvimento de conferências síncronas, ele apresenta os eventos presentes, com apresentação imediata do que está ocorrendo dentro da conferência, suportando, no máximo a persistência desta após a saída de seu último membro, o que pode ser considerado como um suporte a eventos em um passado contínuo, embora não muito adequado, com uma persistência baixa.

Os últimos desenvolvimentos dentro do GroupKit têm vindo na direção de um suporte mais adequado a *wokspace awareness*, com o desenvolvimento de novos *widgets* para isto. Atualmente, há um suporte já preparado, através principalmente das abstrações de programação e dos próprios *widgets*, a interfaces tanto WYSIWIS puras, quanto WYSIWIS relaxadas, embora tanto a escolha por um destes modelos de acoplamento de interface, quanto a própria escolha da metáfora a ser seguida depende essencialmente da aplicação desenvolvida pelo programador, sendo de sua livre escolha.

Quanto à presença, o GroupKit já traz em sua distribuição um *widget* para tanto, o "Participantes" e uma aplicação de conferência, chamada "Post It", que permite a troca de mensagens textuais síncronas entre os participantes de uma conferência.

3.3.9 BSCW (φ)

O BSCW (*Basic Support for Cooperative Work*) *Shared Workspace* é um sistema de armazenamento e recuperação de documentos concebido para o suporte ao trabalho de grupos geograficamente dispersos, particularmente aqueles envolvidos em grandes pesquisas e desenvolvimento de projetos, cujos membros vêm de vários países e organizações.

O BSCW é baseado na noção de *workspace* compartilhado, o qual pode conter vários tipos de objetos, como documentos, tabelas, gráficos, planilhas e links para páginas web [GMD 00a]. O próprio BSCW consiste de um servidor que mantém estes *workspaces* compartilhados e os objetos nestes contidos, os quais são acessados pelos usuários via *web browsers.* Este servidor, desde a versão 2, de junho de 1996, está completamente implementado via interface CGI com o servidor WWW, que chama *scripts* Python, os quais fornecem as facilidades do BSCW. Dentre estas facilidades para o compartilhamento especialmente assíncrono de informações estão o armazenamento e a recuperação destas informações com características como administração de grupos e membros, acesso a meta-informações sobre documentos e membros, serviço de percepção baseado em eventos (*event-based awareness*) e integração com aplicações externas [BEN 98], [GMD 00b].

Cada *workspace* dentro do BSCW contém um certo número de objetos compartilhados e seus membros podem realizar ações como recuperar, modificar e requisitar maiores detalhes sobre estes objetos. O formato da visão do *workspace* é a mesma para cada membro, com uma barra de navegação, um *banner*, uma coluna de botões de ações e a listagem do conteúdo da pasta (*folder*) atual. Este conteúdo é listado diferentemente, dependendo do tipo de visão que o usuário tenha selecionado. Existem três tipos de visão possíveis: a visão de eventos, que mostra à direita de cada objeto um ou mais ícones com os eventos que ocorreram com o objeto; a visão de descrição, que mostra uma descrição dos objetos do *workspace*; e a visão de ação, que mostra uma lista de botões para manipulação de cada objeto, de acordo com os direitos do usuário [BEN 98]. A Fig. 3.16 abaixo traz um exemplo de *workspace* com visão de eventos no BSCW. Estes eventos são sempre relatados para todos os objetos do *workspace*, através desta visão de eventos. A única opção possível ao usuário é suprimir a apresentação de todos os eventos na interface, escolhendo outra visão [SOH 98]. Na visão de eventos, os ícones são agrupados em categorias, como o ícone "novo" ao lado da pasta "screendumps", indicando que a pasta foi incluída no *workspace* após o último ingresso do usuário no sistema, e o ícone ao lado do documento "Boston_Draft", indicando que o mesmo foi alterado, e formam um mecanismo de *awareness* de eventos no passado, pois selecionando-se um destes ícones, uma lista detalhada dos eventos daquele tipo sobre o objeto é mostrada.

Este mecanismo de percepção baseado em eventos atinge eventos no passado e no passado contínuo, com apresentação posterior (quando o usuário ingressa no sistema) e uma alta persistência. Além disso, este mecanismo utiliza essencialmente os próprios objetos compartilhados para conduzir a informação de *awareness*, além de utilizar uma metáfora de sistema, com a idéia de *workspaces* compartilhados sobre um *web browser*.

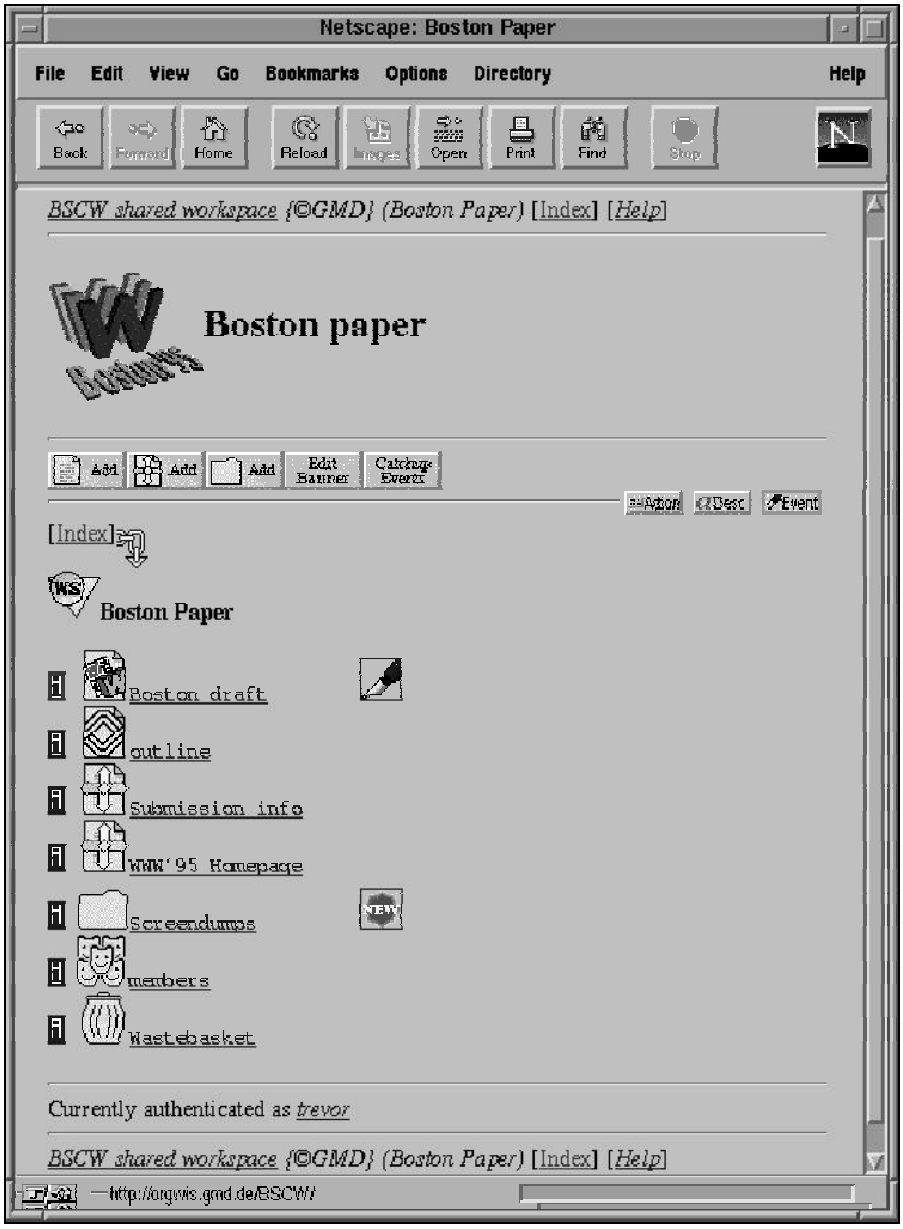


FIGURA 3.16- *Workspace* compartilhado no BSCW [BEN 98]

As três possíveis visões apresentadas pelo BSCW classificam seu mecanismo de percepção como sendo de múltiplas visões, pois permitem visualizações diferentes do *workpace*, permitindo ao usuário escolher aquela que mais se adequar aos seus interesses, mas também podem ser classificadas como desacopladas, pois a visão adotada por um usuário em nada afeta a visão adotada pelos demais.

O BSCW também fornece uma facilidade de check-in/check-out para o gerenciamento de versões, o qual garante que os membros serão alertados do estado atual do documento. Se alguém tentar recuperar um documento em check-out, uma mensagem de notificação é retornada, mostrando quem tem o documento e oferecendo a opção de recuperar o documento assim mesmo, gerando novas versões [BEN 98]. O BSCW permite aos usuários consultar informações sobre estas versões, até que estas sejam explicitamente destruídas por algum usuário do *workspace*, reforçando assim a percepção de eventos no passado e no passado contínuo.

Além destas características, o BSCW também oferece uma ferramenta de comunicação síncrona, um talk implementado em Java, e uma lista com ícones para os participantes ativos no *workspace*, com os quais também é possível estabelecer um link de áudio com um colega [BEN 98]. Com isso, o suporte à percepção também pode ser caracterizado por encarar a presença dos colegas como uma oportunidade, oferecendo ferramentas de comunicação síncronas para aproveitar estas oportunidades. Já as atividades são apresentadas em um "macro-nível", pois somente é possível saber que um documento do *workspace* está em check-out ou que foi alterado durante a ausência do usuário, mas não é possível saber o que foi feito exatamente por um colega. Outro ponto de destaque do suporte à percepção do BSCW é quanto aos papéis, que influenciam na apresentação do *workspace*, especialmente quanto às informações sobre as ações possíveis ao usuário.

3.3.10 WEBDAV (λ)

O WEBDAV _ WWW *Distributed Authoring and Versoning* _ consiste de um grupo de trabalho formado pelo W3C _ *World Wide Web Consortium* _ e pela IETF _ *Internet Engeneering Task Force*, que trata sobre a redação distribuída e a gerência de versões sobre a WWW. Seu principal objetivo consiste em estender o protocolo HTTP, a fim de fornecer uma arquitetura aberta a nível de protocolo para o desenvolvimento de aplicações para autoria distribuída de documentos web, enfatizando a autoria colaborativa de recursos. Para tanto, serão abordados pontos como o controle de acesso, mecanismos de autenticação, e gerência de verões, incluindo mecanismos de *locking*, *check-in/check-out*, fusão automática e acesso a versões anteriores, entre outros [SAL 98], [WEB 99]

O grupo WEBDAV não apresenta, até o momento, maiores preocupações quanto à questão da percepção, embora as conseqüências para sistemas de autoria cooperativa sobre a WWW, se as alterações propostas pelo WEBDAV forem concretizadas, sejam grandes, já que as alterações se darão a nível de protocolo, permitindo um projeto mais simples e uniforme deste tipo de sistema. O mais próximo que o WEBDAV se aproxima da percepção é a proposta de um protocolo para notificação de eventos (*Event Notification Protocol* _ ENP), cujos requisitos estão descritos no Internet Draft deReddy e Fisher [RED 00]. Este documento sugere o ENP como um protocolo da camada de aplicação, que pode ser usado para a notificação distribuída de eventos. Estes eventos são passados pelo ENP dos fornecedores de eventos (*events supplier*) para os consumidores de eventos (*events consumer*) por dois métodos, *push* e *pull*. O primeiro tem o fornecedor tomando a iniciativa da comunicação, enquanto o segundo tem o consumidor tomando esta iniciativa [RED 00].

Alguns dos requisitos apontados por Reddy e Fisher para o protocolo de notificação de eventos são os seguintes [RED 00]: possibilidade de se "registar" para a receber a notificação de certos tipos de eventos e de se ter a filtragem de eventos; a possibilidade de se ter atributos associados às notificações; ter-se uma fila para a entrega destas notificações; ter notificações com ou sem garantia de entrega; ter a possibilidade de especificar parâmetros de Qualidade de Serviço (*Quality of Service* _ QoS); ter-se a notificação mesmo sob *firewall* e também em ambientes de redes seguros. Os consumidores também devem ser capazes de receber eventos de um ou mais fornecedores, e os fornecedores devem ser capazes de fornecer a vários consumidores,

sem necessariamente ter conhecimento da identidade destes , além de suportar modelos de subscrição, com os consumidores requisitando eventos, e modelo de *broadcast*, com os consumidores sendo notificados de eventos mesmo sem estarem escritos para eles.

A abordagem desses requisitos conduz a um protocolo que pode ser usado para o suporte à percepção. Este suporte se resumiria a aspectos como eventos no passado e passado contínuo, pelo uso de filas de entrega e pela própria idéia de controle de versões do WEBDAV, e eventos presentes, com o uso de critérios de QoS, com apresentação posterior e imediata, respectivamente. Todavia, o grupo de trabalho WEBADV ainda se encontra em desenvolvimento e maiores avanços para fornecer meios para o desenvolvimento de um suporte mais adequado à percepção podem ainda ser definidos. Maiores informações podem ser encontradas junto à página da Internet Engineering Task Force [IET 98].

3.3.11 A classificação geral dos mecanismos

Vistas as características de cada um dos sistemas apresentados anteriormente, a Tab. 3.6 resume a classificação destes sistemas e, por conseguinte, dos mecanismos de suporte à percepção por eles implementados. Lembrando que para cada sistema foi relacionado um símbolo e uma cor, que estão distribuídos na Tab. 3.6 apresentada mais adiante. Junto a cada símbolo também pode estar associado um sinal de menos (“-“) para indicar que o suporte à característica não está sendo completo ou adequado, tudo de acordo com o que foi comentado antes nesta seção.

3.4 Considerações finais

Este capítulo discutiu a questão da percepção em sistemas de *groupware*, incluindo o conceito de percepção (*awareness*) e como este conhecimento é importante para que a cooperação alcance seus objetivos de maior produtividade e eficiência. Graças à percepção, os membros de um grupo podem coordenar e estruturar melhor suas próprias atividades, através do conhecimento das atividades dos demais e do grupo como um todo.

Apresentou-se também uma classificação para este domínio, resumindo as principais características para o suporte à percepção dentro de ferramentas de *groupware*. Estas características podem ser divididas em seis questões (o que, quando, onde, como, quem e quanto), sempre analisadas sob os pontos de vista de sistemas síncronos e assíncronos. Sob a luz desta classificação, analisou-se diversos mecanismos de suporte à percepção utilizados em sistemas de diversos tipos, síncronos e assíncronos, variando entre *toolkits* e *frameworks*, como o GroupKit e o COPSE, editores, como o Byzance e o CUTE, sistemas de apoio, como o SISCO, para pré-reuniões, e sistemas mais específicos, como o PortHoles, para indicação, presença e disponibilidade, e outros mais gerais, como o POLIAwaC e o DIVA. Com isso, chegou-se a um resumo significativo das pesquisas no domínio da percepção, útil não só no sentido de organizar as contribuições mais significativas neste domínio, mas também para auxiliar novas pesquisas, apresentando-lhes um panorama geral, destacando o que já foi realizado e mostrando os pontos ainda descobertos, onde maiores trabalhos ainda são necessários para o melhor desenvolvimento da área.

TABELA 3.6 - Classificação de mecanismos de *awareness* encontrados na literatura

		Ambientes Síncronos	***Ambientes Assíncronos***
O que	*Atividades*	micro-nível (α-) (ε-) (η)	macro-nível (δ) (ε) (ϕ) (γ) (φ)
	Papéis	diferenciação das informações de acordo com o papel desempenhado (δ-) (ϕ) (γ-) (φ)	
Quando	*Eventos*	Presente (α) (ε) (η) (λ)	Passado (δ) (γ) (φ) (λ) Passado Contínuo (δ) (ε) (ϕ) (γ) (η-) (φ) (λ)
		Futuro	
	Apresentação	Imediata (α) (η) (λ)	Posterior (δ) (ϕ) (γ) (φ) (λ)
	Persistência / Tempo De Utilidade	Baixa (α) (η)	Alta (δ) (ε-) (ϕ-) (γ) (φ)
Onde	*Espaço*	*workspace awareness* (α-) (ε-) (η)	objetos compartilhados (δ) (ϕ) (φ)
	Metáfora	escritório (ε)	sistema (α-) (δ-) (ϕ) (φ)
Como	*Interface*	WYSIWIS (η)	Desacopladas (δ) (γ) (φ)
		WYSIWIS relaxado (α) (η)	
		Múltiplas Visões (δ-) (ε) (ϕ) (φ)	
	Balanceamento	Filtragem (β) (δ) (λ)	
		Agrupamento (β) (δ)	
Quem	*Presença*	Obrigatoriedade (α) (β) (ε) (η)	Oportunidade (χ) (δ-) (ε) (γ-) (φ)
	Ferramentas De Comunicação	Ferramentas síncronas (α) (β) (ε) (γ) (η) (φ)	Ferramentas assíncronas (β) (δ-) (ε) (γ-)

A análise feita neste capítulo mostrou, por exemplo, que existe um suporte muito tímido à percepção de eventos no passado, mesmo em sistemas assíncronos, onde este seria mais necessário, e também um suporte muito pequeno à questão do agrupamento e filtragem das informações, pontos que serão diretamente atacados por esta dissertação, vindo desta identificação a maior importância deste capítulo para o trabalho.

4 Mecanismo Contextualização: uma Proposta de Suporte à Percepção de Eventos no Passado

Os capítulos anteriores apresentaram os principais conceitos sobre Trabalho Cooperativo Suportado por Computador (CSCW) e sobre percepção (*awareness*). Através destes conceitos, pôde-se observar certas carências no suporte à percepção em ferramentas de *groupware*. Uma destas carências refere-se ao suporte à percepção de eventos no passado, a qual consiste no problema central tratado nesta dissertação. Este capítulo destina-se a descrever este problema, uma vez que sua compreensão é vital para a própria compreensão deste trabalho, e também detalhar a solução que está sendo proposta em linhas gerais.

4.1 O Problema: Percepção de Eventos no Passado

No capítulo anterior, observou-se que o suporte a eventos no passado, com persistência alta e apresentação posterior, é limitado a alguns poucos sistemas. No geral, há ausência de um suporte adequado a estas características na maioria das ferramentas de *groupware* estudadas. Além disso, como bem observou Veertegall et al. [VEE 97], a baixa aceitação de ferramentas como videoconferências, com recursos de áudio e vídeo, indica que importantes aspectos da comunicação humana podem estar sendo negligenciados e a percepção é certamente um destes aspectos. Esta ausência de suporte é, na verdade, a falta de conhecimento sobre o que ocorreu dentro do grupo e pode provocar uma série de problemas e inconvenientes, prejudicando o andamento dos trabalhos, à medida que as decisões e fatos ocorridos em um momento passado afetam o trabalho presente. Sem este conhecimento do passado, o grupo corre o sério risco de repetir erros e decisões equivocadas, já realizadas ou discutidas, ou ainda de tomar caminhos que já foram identificados como improdutivos.

Outra situação, na qual o trabalho do grupo é prejudicado pela ausência de um suporte adequado à percepção de eventos no passado, é quando o grupo necessita de mais de uma sessão para atingir os seus objetivos, ou ainda, quando precisa de um período de tempo longo para isto. Nestes casos, a memória das atividades e decisões tomadas nas sessões anteriores ou ao longo do período é de vital importância para evitar que atividades redundantes ou conflitantes sejam realizadas, causando uma perda de produtividade dentro do grupo. O problema se agrava ainda mais quando se tem um ambiente assíncrono, onde os membros não precisam necessariamente trabalhar em um mesmo intervalo de tempo. Nestes ambientes, os membros precisam ser atualizados sobre o que foi realizado durante sua ausência sempre que retomarem suas atividades junto ao sistema.

Considerando-se, por exemplo, a Fig. 4.1 abaixo. Nesta, um grupo trabalha ao longo de um período de tempo, indicado pelo eixo localizado na base da figura. Durante parte deste período, um membro, na parte superior da figura, permanece afastado das atividades do grupo, participando somente de um primeiro momento, onde todos os membros têm a oportunidade de estarem trabalhando em um mesmo período de tempo e dentro deste

dividem as tarefas nas atividades “A”, “B” e “C”. Em um segundo momento, este membro localizado no alto da figura se afasta, enquanto os demais colegas seguem trabalhando, inicialmente com apenas dois membros trabalhando simultaneamente, e em um terceiro momento, quando três deles conseguem um mesmo intervalo de tempo para trabalhar. Dentro do segundo momento, um dos membros permanece trabalhando sobre a atividade “B”, enquanto o outro trabalha sobre uma nova atividade “D”. Já no terceiro momento, os três membros presentes concordam que a atividade “A” não é mais necessária, removendo-a do conjunto de atividades. A partir desse momento, o primeiro membro, que esteve ausente nos dois últimos encontros, precisará necessariamente de algum mecanismo que lhe forneça percepção sobre os eventos passados para que este possa retomar seu trabalho junto ao grupo sem recair em atividades redundantes ou conflitantes, como seria o caso se ele retomasse a execução da atividade “A”.

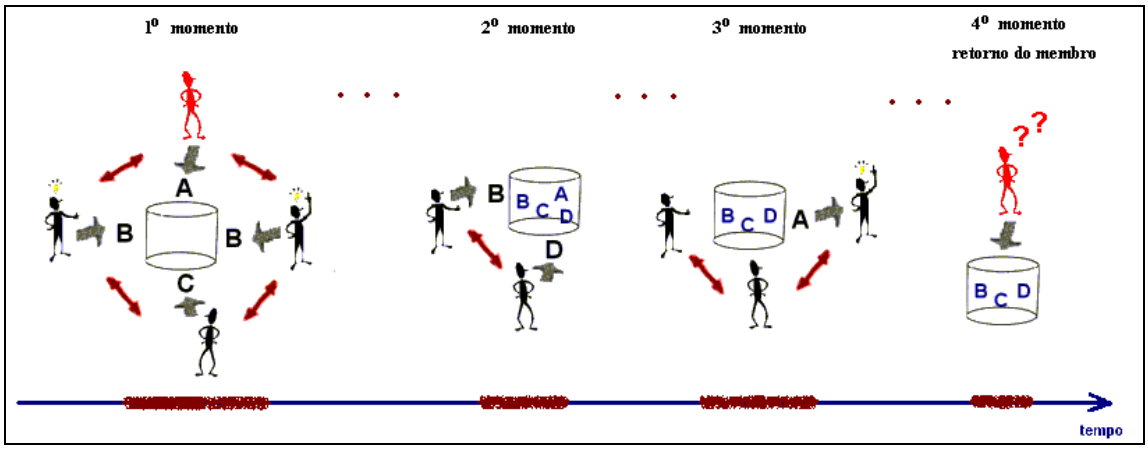


FIGURA 4.1 - Exemplo de necessidade de percepção

O desenvolvimento de software é outra situação onde este tipo de problema pode ser encontrado. Segundo Venners [VEN 98], este processo de desenvolvimento tende a ser interativo, pois envolve o *feedback* contínuo de várias partes e caótico, pois membros do time vão e vêm, requisitos mudam e projetos inteiros podem ser cancelados. Neste caso, manter um histórico sobre as atividades passadas do grupo é vital para que o processo de desenvolvimento progrida e não repita velhos erros, nem atividades já realizadas, mesmo quando há troca de membros da equipe. Havendo um mecanismo de suporte à percepção dos eventos ocorridos no passado, um membro novo ao grupo poderá se beneficiar deste mecanismo para ficar a par do que já foi realizado dentro da equipe, ou ainda no caso de alterações nos requisitos, os próprios membros da equipe podem usar este mecanismo para rever o que foi feito e a partir daí decidir os rumos que o trabalho deve tomar para satisfazer os novos requisitos.

Nestes ambientes de desenvolvimento, é necessário fornecer para todos os membros “a noção do contexto de suas atividades dentro do contexto geral do processo, para que consigam perceber o andamento das atividades sendo realizadas por outros membros e para compreender como resultados gerados pelas atividades alheias podem ser conjugados aos seus para chegarem mais rapidamente ao produto final” [DIA 98]. Este contexto é dado pelo suporte à percepção e sua ausência nestes ambientes pode significar um produto final, software, de baixa qualidade, ou ainda, um ciclo de produção ineficiente.

Também é possível citar a questão da edição cooperativa de documentos, ou ainda, a edição de hiperdocumentos. Segundo Borges et al. [BOR 95], “criar

hiperdocumentos complexos e com grande volume de dados requer um trabalho coordenado e, acima de tudo, uma grande interação entre vários autores". Além disso, Tammaro et al. [TAM 97] tratam a edição cooperativa justamente como o processo onde dois ou mais autores trabalham juntos para criar um documento complexo. Agora, imaginando que um destes autores seja obrigado a se ausentar por um longo período, seja por razões pessoais, seja por problemas em sua conexão de rede, como ficará este membro depois de seu retorno ao grupo, se não houver um mecanismo adequado que o coloque a par do que foi realizado em sua ausência? Este membro ficará perdido, sem condições de trabalho, sem um suporte adequado à percepção de eventos no passado, com persistência alta (para suportar este período de ausência) e apresentação posterior (quando o membro retornar as suas atividades em grupo). Isto porque ele precisará reler todo o hiperdocumento para assim conseguir observar as alterações realizadas, baseado em sua própria memória, ou ainda recorrer aos colegas, através de canais de comunicação informal para perguntar-lhes o que foi realizado. Este certamente não é o melhor caminho para ficar a par destas alterações e torna o retorno deste membro ausente mais difícil, prejudicando seu desempenho junto ao grupo.

Assim, o foco central desta dissertação é o suporte à percepção de eventos no passado, com persistência alta e apresentação posterior, em especial para ambientes com características assíncronas, onde os membros não precisam necessariamente trabalhar em um mesmo intervalo de tempo e em ambientes onde são necessárias várias sessões para que o objetivo seja atingido. O que se pretende atacar aqui é, então, o problema do membro que é obrigado a se ausentar do grupo (como o da Fig. 4.1), buscando facilitar o seu reingresso no ambiente cooperativo. Para tanto, será fornecido a este membro o conhecimento sobre as atividades passadas que foram realizadas durante a sua ausência, que podem influenciar em seu trabalho. Convém ressaltar que o objetivo deste trabalho são somente estas atividades que já aconteceram dentro do grupo e não aquelas atividades que estão ocorrendo agora, como, por exemplo, um movimento de mouse dentro do *workspace* compartilhado, nem mesmo aquelas que iniciaram em um momento passado, mas ainda não foram concluídas, como, por exemplo, a edição de um capítulo dentro de um documento compartilhado.

A idéia desta dissertação é única e exclusivamente atacar a falta de percepção sobre as atividades que já foram concluídas dentro do trabalho em grupo, desenvolvendo um mecanismo que forneça a percepção destas, e deixar apenas como possibilidade de trabalhos futuros o suporte à percepção de atividades em um passado contínuo. Isto significa que somente aquelas atividades que já foram concluídas serão tratadas aqui. Por exemplo, a edição de um capítulo em um documento compartilhado por um grupo de usuários. Enquanto esta edição não for concluída, esta atividade será considerada um evento no "passado contínuo", uma vez que ela começou em um momento passado, mas ainda não foi concluída ou sobreposta. Ela somente será de interesse deste trabalho quando o processo de edição do capítulo for concluído. Somente neste momento esta atividade será considerada um evento realmente no "passado", seguindo a classificação proposta no capítulo anterior. É importante frisar este enfoque para somente eventos no "passado", e não os no "passado contínuo", pois diversas decisões de projeto foram tomadas tendo em vista esta situação específica. O suporte a atividades em um passado contínuo, ou seja, a atividades que ainda não foram concluídas, mesmo tendo iniciado em um momento no passado, não é tratado dentro deste trabalho, sendo citado como uma possível proposta de trabalho futuro.

Para solucionar, então, o problema proposto, esta dissertação sugere a criação de um mecanismo de contextualização flexível, na forma de um *framework*. Este mecanismo, descrito em linhas gerais na próxima seção, ataca diretamente às questões "quando" e "como" da classificação apresentada no capítulo anterior, buscando oferecer suporte à percepção de "Eventos" no "Passado", com "Apresentação" das informações "Posterior" à ocorrência dos eventos geradores, com uma "Persistência", ou "Tempo de Utilidade", "Alta", para suportar um certo período de ausência dos membros participantes do grupo. O mecanismo ainda pretende atingir a questão "Como", mas especificamente o "Balanceamento" das interfaces, através de uma "Filtragem" das informações de percepção, a fim de evitar a sobrecarga cognitiva sobre os usuários. A Tab. 4.1 abaixo mostra as questões abrangidas pelo mecanismo em destaque sobre um trecho da classificação apresentada no capítulo anterior.

TABELA 4.1 - Classificação do mecanismo de monitoramento e contextualização

		Ambientes Síncronos	***Ambientes Assíncronos***
Quando	*Eventos*	Presente	**Passado**
			Passado Contínuo
		Futuro	Futuro
	Apresentação	Imediata	**Posterior (a critério do usuário)**
	Persistência/Tempo de Utilidade	Baixa	**Alta (critério de caducidade)**
Como	*Interface*	WYSIWIS	WYSIWIS relaxado
		WYSIWIS relaxado	Múltiplas Visões
		Múltiplas Visões	Desacopladas
	Balanceamento	**Filtragem**	
		Agrupamento	

4.2 Proposta de Mecanismo de Contextualização

Para solucionar o problema descrito na sessão anterior, este trabalho propõe o desenvolvimento de um mecanismo que monitore as atividades realizadas pelos participantes dentro do grupo para posteriormente fornecer-lhes o contexto das atividades realizadas por seus colegas. Este monitoramento é necessário, pois somente com o registro das atividades passadas é possível fazer a contextualização dos participantes sobre a ocorrência destas atividades. Dessa forma, este mecanismo ataca a situação descrita na Fig. 4.1, onde um membro se ausenta por um período e precisa ser contextualizado sobre atividades realizadas no momento de seu retorno.

Este mecanismo também tem por objetivo fornecer a percepção de eventos no passado, com persistência alta e apresentação posterior de forma flexível, uma vez que existe hoje uma grande variedade de ferramentas de *groupware* já desenvolvidas que não contam com este suporte à percepção. Exemplos destas são o CUTE e o Byzance, que oferecem um suporte muito limitado a esta percepção, conforme pôde ser observado no Capítulo 3. Os poucos casos que apresentam este tipo de suporte, como a ferramenta POLIAwaC, incluída no ambiente POLITeam (também vista no Capítulo 3), normalmente

são muito especializados, impossibilitando sua generalização e implantação junto a outras ferramentas de *groupware*. Dessa forma, o desenvolvimento de um mecanismo de percepção flexível, capaz de ser integrado a outras ferramentas de *groupware*, fornecendo-lhes um suporte que antes lhes era ausente, é de grande utilidade e constitui a maior contribuição deste trabalho.

Esta possibilidade de integração é extremamente vantajosa do ponto de vista de custos, pois é menos onerosa do que seria a reconstrução do *groupware* visando a inclusão deste suporte à percepção, uma vez que todo o *racionale* do processo se limitaria aos pontos de adaptação do mecanismo ao *groupware*, não sendo necessário ao desenvolvedor deste último (*groupware*) conhecimentos profundos sobre a questão da percepção. Isto implicaria em uma economia em termos de horas de aprendizagem (conhecimento sobre *awareness*) e desenvolvimento (construção do suporte à percepção), as quais são reduzidas ao aprendizado do mecanismo e adaptação deste. Assim, este mecanismo aparece como um auxiliar chamado pelo *groupware* para atender suas necessidades de percepção de eventos no passado, como indica o esquema apresentado na Fig. 4.2 abaixo, onde o usuário aparece interagindo com o *groupware* e este, por sua vez, interagindo com o mecanismo de monitoramento e contextualização.

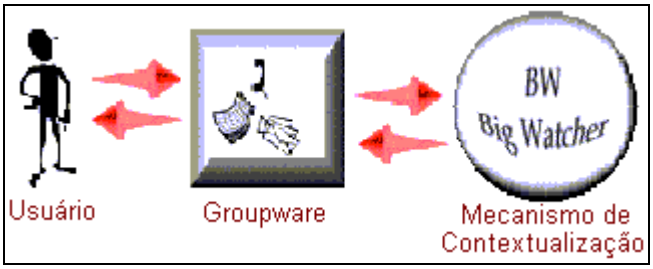


FIGURA 4.2 - Esquema geral do mecanismo de contextualização proposto

Para chegar a este nível de flexibilidade desejado, o mecanismo aqui proposto parte da idéia simples de um esquema de notificação de eventos, ou *event-based awareness*. O uso deste conceito de notificação de eventos e filtros em CSCW e no fornecimento de *awareness* já vem sendo utilizado por vários autores (ver, por exemplo, Mariani [MAR 97], Belassai et al. [BEL 95], Parasowith et al. [PAR 99], Mansfield et al. [MAN 99] e Nomura et al. [NOM 98]), e tem se apresentado como uma forma simples, mas poderosa para o suporte à percepção.

Dentro deste trabalho, este conceito de notificação de eventos é utilizado para o mapeamento das atividades do grupo e posterior contextualização dos participantes sobre as mesmas da seguinte forma: cada tipo de atividade que deverá ser monitorada é interpretada pelo mecanismo como um tipo de evento. Quando uma atividade de um determinado tipo ocorre, um evento do tipo correspondente é enviado ao mecanismo que o armazena para posteriormente fazer a contextualização dos participantes, através da notificação. Todos estes tipos de eventos, que correspondem aos tipos de atividades observadas, são registrados pelo *groupware* dentro do mecanismo de contextualização de acordo com seus interesses. Dessa forma, o mecanismo aqui proposto tem seu primeiro passo em direção à flexibilidade desejada dentro desta idéia de *event-based awareness* nestas três etapas que compõem o ciclo de registro, ocorrência e notificação de eventos, apresentado na Fig. 4.3 abaixo, com sua ordem normal de execução indicada.

Convém ressaltar que o conceito de evento utilizado neste trabalho é equivalente a uma atividade ocorrida e concluída e não deve ser confundido com outros conceitos utilizados junto ao termo "evento", como os citados em Leticia et al. [LEI 99] e Wahl e Rothermel [WAH 94]. O conceito de evento utilizado aqui é discutido mais detalhadamente no próximo capítulo, quando é apresentado o modelo de dados e a descrição formal destes eventos através de um intervalo de Allen (ver Allen [ALL 83]).

FIGURA 4.3 - Ciclo de registro-ocorrência-notificação

O segundo passo para esta flexibilidade foi a construção do mecanismo aqui proposto na forma de um *framework*. Um *framework* consiste em uma mini-arquitetura que fornece uma estrutura geral e o comportamento para uma família de abstrações de software. Um *framework* não é uma aplicação completa, mas aplicações podem ser construídas sobre um ou mais *frameworks*. Além disso, quando se utiliza um *framework* normalmente só se implementa algumas funções ou se especializa algumas classes, através dos "*hot spots*", implementados comumente através de *callbacks*, polimorfismo ou delegação, permitindo assim que o *framework* seja estendido, adaptado e até combinado com outros *frameworks* [APP 99], [GAM 94].

Dessa forma, a construção do mecanismo de contextualização na forma de um *framework*, batizado de BW _ *Big Watcher* _ devido as suas características de monitoramento, fornece uma estrutura geral para o fornecimento de percepção de eventos no passado e um comportamento dentro da idéia de *event-based awareness*, discutida anteriormente. Além disso, dentro do próprio *framework* já estão identificados os "*hot spots*", pontos onde o *framework* deve ser adaptado e estendido, via *callback* ou polimorfismo, a fim de adequar-se melhor ao *groupware* base, onde está sendo incluído. O Capítulo 6, mais adiante, discutirá em detalhes o *framework* construído, aqui é apresentado somente um esboço geral do mesmo.

Em linhas gerais, o *framework* BW pode ser dividido em três partes, respeitando uma estrutura de três camadas definida para o mecanismo, ilustrada na Fig. 4.4: interface, controle e armazenamento. A camada de armazenamento é responsável pela manutenção

em disco dos eventos ocorridos e registrados no mecanismo, além de outras informações necessárias. A camada de controle é responsável pelo processamento propriamente dito, controlando tanto o monitoramento, incluindo o registro e a ocorrência de eventos, quanto à contextualização dos participantes, através da notificação dos eventos submetidos a filtros de preferências (*profiles*) dos usuários. A camada de interface é responsável pelo contato com o usuário, principalmente pela apresentação das informações de *awareness* selecionadas pelo participante.

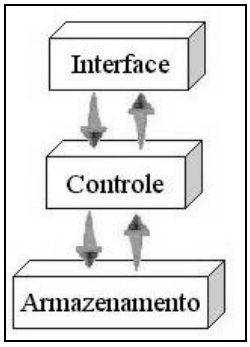


FIGURA 4.4 - Camadas do mecanismo de contextualização

Com estas camadas, o mecanismo divide a execução do ciclo de registro, ocorrência, e notificação, com cada camada fazendo sua parte no ciclo. A camada de controle trata o pedido de registro de eventos que devem ser monitorados e também o aviso de ocorrência destes eventos e, em ambos os casos, repassa estas informações para a camada de armazenamento que as organiza e guarda em disco. A camada de interface avisa a camada de controle sobre a presença de um usuário, para que esta última, com base nos *profiles* do usuário, solicite à camada de armazenamento os eventos ocorridos que são de interesse deste usuário, para então repassá-los de volta à interface, que finalmente faz a notificação do usuário. A Fig. 4.5, a seguir, exemplifica através de um esquema, como ocorre o processo de notificação entre as camadas.

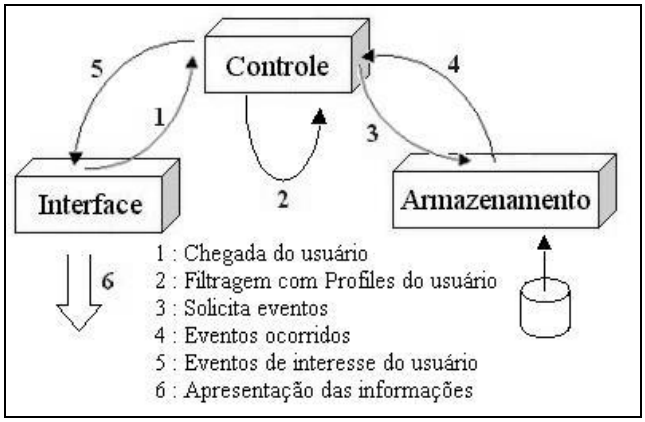


FIGURA 4.5 - A notificação entre as camadas

Com relação ao processo de adaptação do BW ao *groupware*, em cada uma das camadas dentro do *framework* existem pontos (os "*hot spots*"), os quais podem sofrer adaptações. Em termos de armazenamento, as classes que descrevem o meio onde os

dados serão mantidos, se em um banco de dados ou em um sistema de arquivos puro, etc, podem ser especializadas a fim de prover novas estratégias de armazenagem para o mecanismo, ou ainda podem ter certos atributos alterados para melhor refletirem a realidade do *groupware*, conforme será visto em detalhes no Capítulo 6. Quanto ao controle, as classes que descrevem os eventos também podem ser especializadas para melhor descrever eventos específicos da aplicação de *groupware* os quais, via polimorfismo, são usados dentro do mecanismo. Ainda dentro do controle, a classe que descreve os papéis também pode ser especializada, da mesma forma que os eventos, para representar o modelo hierárquico adotado pelo grupo.

Esta representação dos papéis por uma classe abstrata que pode ser especializada para representar o modelo hierárquico do grupo é o terceiro passo para a flexibilização do mecanismo de contextualização. Através deste passo, o mecanismo não precisa se ater a um modelo hierárquico pré-definido, podendo ser utilizado em várias ferramentas de *groupware*, com diferentes modelos hierárquicos e mesmo assim ser capaz de representá-los internamente.

Fato semelhante ocorre com o modelo cognitivo. Este modelo consiste nos passos necessários para que o objetivo seja atingido, conforme foi visto no Capítulo 2. É altamente dependente do tipo de aplicação, se é um *groupware* de edição cooperativa, ou um sistema de suporte à decisão, ou ainda uma conferência, etc, e do próprio grupo, que é o principal responsável pela escolha desse modelo. Cada grupo pode ter um modelo cognitivo particular, adaptado aos seus interesses e ao seu estilo de trabalho. Desta maneira, a não adoção de um modelo cognitivo fixo para o mecanismo de contextualização consiste em uma garantia de flexibilidade, pois o contrário significaria engessar o mecanismo a um tipo único de *groupware* e principalmente de grupo, colocando abaixo todo o objetivo e os esforços em busca de flexibilidade para este trabalho.

Com isso, o *framework* BW não adota nenhum modelo para o ordenamento das atividades, assumindo que o *groupware* base o faça, de acordo com o modelo cognitivo por este adotado. O *framework* somente aceita os eventos indicadores de ocorrência de uma atividade, sem questionar sua ordem: quando a atividade é concluída, o *groupware* alerta o mecanismo sobre a ocorrência do evento correspondente e este último apenas considera que o evento aconteceu na ordem adequada com o modelo cognitivo do *groupware*. O *framework* BW assume apenas que, se a atividade não fosse de interesse do *groupware* ou estivesse em desacordo com o modelo cognitivo, este *groupware* não o alertaria sobre a ocorrência do respectivo evento.

Já quanto à interface, o uso de "*hot spots*" dentro do *framework* também é intenso. A interface com o usuário final constitui um problema para o mecanismo, uma vez que ela constitui o meio pelo qual o usuário recebe todas as informações da percepção. Se esta interface for inadequada, toda a eficácia do mecanismo fica comprometida. Idealmente, esta interface deve ser otimizada para os tipos de atividades que serão observadas e para as quais é fornecida a percepção. Deve também, preferencialmente, estar embutida dentro da área de trabalho do usuário, de modo que ele não precise sair à procura das informações de *awareness*, mas sim que elas estejam ao seu alcance. Em outras palavras, pode-se afirmar que a interface é totalmente dependente do *groupware*, das atividades monitoradas e da própria interface do *groupware*, uma vez que o usuário, além de precisar das informações ao seu alcance, não deveria sentir diferença entre as informações vindas do mecanismo de percepção e do próprio *groupware*, mas apenas ver tudo como sendo o *groupware*. Esta dependência coloca o mecanismo de contextualização em uma situação delicada, já que este busca ser flexível e não preso a nenhuma ferramenta de *groupware* específica, mas parte importante de sua eficácia depende do *groupware* utilizado como base.

Para resolver esta questão, o mecanismo aqui proposto usa de dois artifícios. O primeiro é o já mencionado uso de "*hot spots*" dentro do *framework* BW. Estes "*hot spots*" usam a idéia do polimorfismo para a interface com o usuário, formando-a essencialmente através de classes abstratas. Estas classes descrevem a interface como sendo construída de pequenos elementos, como os *widgets*, os quais estão aptos a receber as informações de *awareness*. Entretanto, estas classes são todas abstratas, nenhuma implementação lhes é fornecida. Toda a implementação deve ser fornecida pelas subclasses, as quais são utilizadas dentro do mecanismo via polimorfismo, mas são construídas pelo desenvolvedor do *groupware*. Este desenvolvedor é a pessoa que conhece (e possivelmente ajudou a construir) a implementação do *groupware* e agora está incluindo neste um suporte à percepção através do *framework* BW. É este desenvolvedor que deverá especializar as classes abstratas que compõem a camada de interface (bem como os outros *hot spots* nas demais camadas), para assim construir a interface com o usuário propriamente dita, de modo que esta seja capaz de receber as informações de percepção e ainda estar perfeitamente introduzida dentro do ambiente do *groupware*.

Todavia, para prevenir que a interface gerada por este desenvolvedor não seja adequada, o mecanismo aqui proposto faz uso de seu segundo artifício para interface: um pequeno guia de construção da interface, uma espécie de *guideline*. Este guia, que será apresentado na íntegra no Capítulo 6, é composto por uma série de orientações para conduzir o desenvolvedor na construição de uma interface clara, prática, integrada ao *groupware*, de fácil assimilação e que não sobrecarregue o usuário com muitas informações. Deve-se, porém, ressaltar que a construção deste guia não é o objetivo principal deste trabalho, mas uma necessidade que surgiu durante o seu desenvolvimento. Este guia não tem, de forma alguma, a pretensão de ser um guia completo para a construção de interfaces para o suporte a *awareness*, mas apenas procura ser um conjunto de sugestões para auxiliar a construção destas interfaces. A proposta de um guia completo com este objetivo seria, por si só, um outro trabalho de mestrado e não é o objetivo desta dissertação posar como tal trabalho.

Esta preocupação com a interface remete à questão "onde" discutida no capítulo anterior, mas especificamente quanto ao "balanceamento" desta interface. Esta preocupação reside essencialmente na quantidade de informação de percepção apresentada para o usuário. Estas informações de percepção devem ser em quantidade tal que não deixem o usuário sem informações importantes para sua contextualização, nem o sobrecarreguem com muitas, demandando atenção em demasia. Para tanto, o desenvolvedor deverá seguir as sugestões presentes no guia apresentado no Capítulo 6 para construir uma interface com o usuário que procure apresentar estas informações de forma reduzida, fazendo com que a assimilação destas seja mais rápida, não o obrigando a procurar exaustivamente por estas informações, nem demandando atenção em demasia do usuário, pontos que poderiam vir a prejudicar o decorrer de seu trabalho.

Além de apresentar o guia para auxiliar o desenvolvedor na produção de uma interface balanceada, o mecanismo de contextualização dispõe ainda do conceito de filtros, os *profiles* do usuário. Cada usuário, através dos *profiles*, indica quais tipos de eventos, representando quais tipos de atividades são de seu interesse, permitindo que a camada de controle faça uma filtragem nos eventos ocorridos, mantidos pela camada de armazenamento e passando para a interface somente aqueles que são do interesse do usuário.

Estes *profiles*, representados dentro do *framework* BW por classes relacionadas às classes de participantes (usuários) e papéis, armazenam uma lista de tipos de eventos que são considerados "interessantes" e dividem-se em três tipos, de acordo com seus

relacionamentos. O primeiro tipo é relacionado diretamente a cada usuário com suas preferências pessoais, representando os interesses particulares do usuário. O segundo é relacionado a cada papel que pode ser desempenhado pelos participantes dentro do grupo, representando as informações que são importantes para os participantes, quando estes estiverem desempenhando determinado papel, enquanto o último, relacionado tanto com o usuário quanto com os papéis que podem ser exercidos por ele, contendo as opções pessoais deste usuário quando ele estiver desempenhando um papel específico. As informações contidas nestes três tipos são sempre combinadas (através de um "e" lógico) para formar o filtro aplicado ao conjunto de eventos disponíveis, resultando em um subconjunto formado somente por aqueles tipos de eventos que representam os tipos de atividades que são de interesse pessoal do usuário ou do papel que por ele está sendo desempenhado no momento, conforme mostra o esquema apresentado na Fig. 4.6 abaixo.

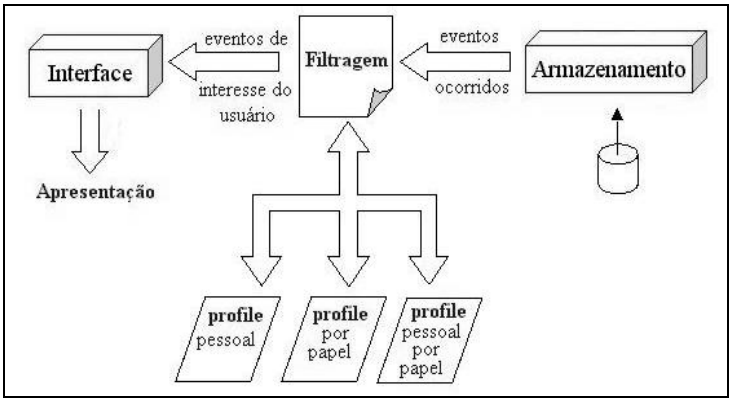


FIGURA 4.6 - Filtragem utilizando *profiles*

Através do guia para auxílio à construção de interface e deste esquema de filtragem através dos *profiles*, minimiza-se a possibilidade de sobrecarga de informações sobre o usuário, limitando-as a somente aqueles tipos de eventos que correspondem aos tipos de atividades que são de seu interesse pessoal e profissional, e controla-se a questão da eficácia do mecanismo, dependente da interface, sem torná-lo dependente também do *groupware* utilizado como base, mantendo assim a flexibilidade objetivada para o mecanismo.

4.3 Considerações Finais

Este capítulo apresentou o problema abordado por esta dissertação, a percepção de eventos no passado, com persistência alta e apresentação posterior, principalmente em ambientes assíncronos, onde os membros não precisam trabalhar em um mesmo momento ou são necessárias várias sessões para atingir o objetivo. Apresentou-se também um esboço da proposta de solução, a qual envolve a criação de um mecanismo de monitoramento e contextualização dos participantes em um trabalho em grupo, através da criação do *framework* BW, projetado para ser flexível o bastante a ponto de ser adaptado e incluído em mais de uma ferramenta de *groupware*. Todas as principais questões que

envolvem a construção deste mecanismo foram apresentadas em linhas gerais neste capítulo, fornecendo assim uma visão panorâmica da solução proposta. Destacou-se principalmente o ciclo registro – ocorrência – notificação, a construção do *framework* BW, sua divisão em três camadas e o processo de filtragem através de *profiles*.

A partir deste momento, cada uma destas questões será cuidadosamente tratada dentro dos capítulos seguintes, fornecendo então uma visão detalhada de cada parte da solução apresentada. Em primeiro lugar, o Capítulo 5 apresentará a arquitetura geral do mecanismo, com suas três camadas e o modelo de dados utilizado pela camada de armazenamento para guardar as informações. Em seguida, o Capítulo 6 apresenta o *framework* BW em detalhes, com sua formação e possibilidades de adaptação e ampliação e ainda a questão da interface, com o guia de recomendações para a especialização das classes abstratas da camada de interface.

5 Arquitetura e Modelo de Dados

O capítulo anterior apresentou uma visão geral do mecanismo de contextualização introduzido neste trabalho. Foi discutido o problema atacado, ausência de suporte à percepção de eventos no passado, e apresentada uma visão geral da proposta de solução, a construção do *framework* intitulado "BW". Partindo desta visão geral exposta no capítulo anterior, parte-se agora para a descrição minuciosa desta solução. Neste capítulo são colocados em detalhes a arquitetura geral adotada pelo mecanismo e o seu modelo de dados, responsável pela manutenção das informações geradas e utilizadas por este. Sobre esta arquitetura e este modelo de dados foi construído o *framework* BW, que será apresentado também em detalhes no próximo capítulo.

5.1 Arquitetura Geral

O objetivo geral deste trabalho consiste no desenvolvimento de um mecanismo de suporte à percepção de eventos no passado, por vezes genericamente chamado de mecanismo de contextualização, flexível o bastante para que pudesse ser incluído em mais de uma ferramenta de *groupware*. Para chegar nesta flexibilidade desejada, este mecanismo tem suas bases calcadas sobre uma arquitetura geral dividida em três camadas. Estas três camadas, apresentadas na Fig. 5.1, dividem entre si todas as tarefas, sendo cada uma delas responsável por parte do processamento.

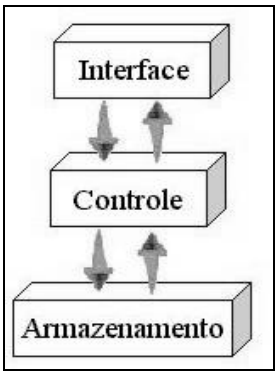


FIGURA 5.1 - Arquitetura geral do mecanismo

A primeira camada, denominada "Interface", é responsável pelo contato com o usuário, fazendo a apresentação das informações de *awareness*. A segunda camada, chamada "Controle", é a camada "pensante" propriamente dita, controlando os eventos e as informações geradas pelo monitoramento das atividades, bem como organizando as informações para a contextualização do usuário. A última camada, "Armazenamento", é a responsável pela manutenção destas informações coletadas em meio permanente.

A vantagem do uso desta arquitetura em três camadas reside justamente na divisão de responsabilidades. Cada camada fica responsável por partes específicas do processamento, atendendo a demanda das demais. Desta forma, cada camada fornece um conjunto bem definido de serviços de acordo com suas responsabilidades. Estes serviços são conhecidos por suas camadas vizinhas que podem solicitá-los de acordo com suas necessidades. Um exemplo destes serviços é o registro de um tipo de atividade a ser monitorada. Este é um serviço oferecido pela camada de controle, a qual é responsável pelo controle das informações que deverão surgir deste monitoramento. Conhecidos os serviços oferecidos por cada camada, estas passam a ser independentes das demais à medida que podem variar internamente sem que as outras percebam. Com isso, a própria complexidade do mecanismo fica dividida entre estas três camadas, uma vez que cada trecho pode ser projetado separadamente dos demais, e trechos com funções semelhantes encontram-se agrupados de acordo com suas responsabilidades dentro das camadas.

Esta arquitetura possui semelhanças com o modelo MVC _ *Model – View – Controller*, presente no SmallTalk. O MVC prevê a divisão das classes em três: *Model*, *View* e *Controller*, com a finalidade de separar a representação visual (*View*) dos dados propriamente ditos (*Model*). Nesta divisão, a classe "*Model*" é o objeto da aplicação, responsável pela manutenção destes dados. As classes "*View*" são apenas responsáveis pela representação gráfica destes dados na tela. Entre as duas estão as classes "*Controller*", que definem como a interface do usuário reage as suas entradas [GAM 94].

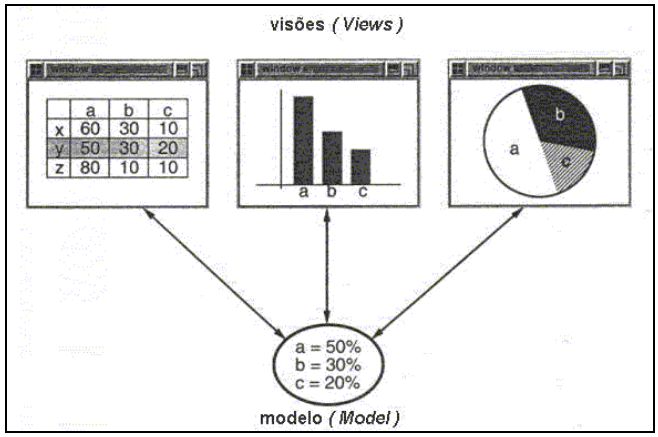


FIGURA 5.2 - Apresentação de um conjunto de dados em 3 visões [GAM 94]

A vantagem no uso do modelo MVC reside na independência entre os dados (*Model*) e a sua visualização (*View*), como mostra a Fig. 5.2 baseada de Gamma et al. [GAM 94], onde um mesmo conjunto de dados é apresentado de três formas diferentes. Esta vantagem também foi procurada pelo mecanismo aqui proposto, motivo pelo qual buscou-se no conhecido modelo MVC a inspiração para esta divisão da arquitetura em três camadas. A camada de interface, assim como as classes de "*View*", fica responsável apenas pela apresentação dos dados, desconhecendo o restante do processo. A camada de armazenamento lida somente com os dados, assim como as classes "*Model*", mas trabalha com estes dados em estado puro, armazenando-os em meio permanente e desconhecendo o processamento que é feito sobre os mesmos antes de armazená-los. Por fim, a camada de controle trata e gera estes dados, desconhecendo como eles são mantidos em disco e como são apresentados ao usuário.

Esta divisão foi também mantida na construção do *framework* BW, descrito no próximo capítulo, onde as classes foram divididas em pacotes buscando respeitar esta estrutura de camadas. Com isso, frisando novamente, a complexidade na construção destes pacotes foi reduzida, uma vez que a área de concentração dos pacotes estava bem definida pela camada em que se encontravam. O uso desta arquitetura em camadas também permitiu o projeto separado de cada uma delas, permitindo por conseqüência, que cada pacote fosse projetado em separado, centrando o foco em suas responsabilidades e nos serviços que devem ser fornecidos pela camada. Convém mencionar que não se seguiu à risca o modelo MVC, mas se buscou apenas inspiração neste para a separação entre dados, visualização e controle, com vistas à simplificação do projeto, de modo que o leitor mais acostumado ao modelo MVC do SmallTalk poderá notar diferenças entre o modelo e a arquitetura adotada neste trabalho, o que é natural, pois este modelo serviu exclusivamente de inspiração para o projeto desta arquitetura e do *framework* BW.

Dessa forma, a camada de interface, responsável pela visualização dos dados, oferece serviços de notificação e comunicação com o usuário. Através destes serviços, as informações são apresentadas ao usuário final sem que as demais camadas precisem conhecer detalhes do projeto e implantação desta interface, isolando nesta camada todo o processo de apresentação gráfica. Já a camada de armazenamento, que implementa o modelo de dados, fornece serviços de manutenção das informações em disco, escondendo das demais camadas como esta manutenção está sendo feita, podendo ser localmente, utilizando banco de dados ou sistema de arquivos, ou via rede, que para as demais camadas o fornecimento do serviço ainda será transparente.

A camada de controle é a que mais serviços fornece às demais, por ser a camada responsável pelo controle das informações dentro da idéia de *"event-based awareness"*, implantada neste mecanismo pelo ciclo de registro – ocorrência - notificação comentado no capítulo anterior. A camada de controle recebe e trata as requisições de registro dos tipos de eventos, correspondentes aos tipos de atividades que serão monitoradas, pois é dela a responsabilidade de tratar a ocorrência destes eventos e usar dos serviços da camada de armazenamento para a manutenção definitiva destas informações. Através do resgate posterior destas informações mantidas pela camada de armazenamento, a camada de controle pode realizar as operações de filtragem de acordo com os interesses dos usuários, definidos por seus *profiles*, para então solicitar à camada de interface que realize o processo de notificação, como mostra a seqüência apresentada na Fig. 5.3

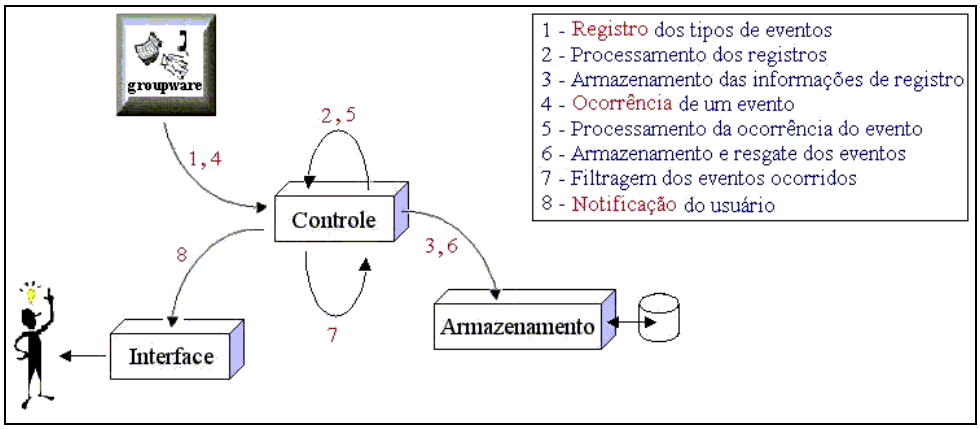


FIGURA 5.3 - Serviços da camada de controle

5.2 Distribuição da Camada de Armazenamento

A arquitetura definida em três camadas bem divididas permite que cada uma delas possa variar internamente sem que as demais percebam esta variação. Com isso, a camada de armazenamento, responsável pela manutenção dos dados, pode optar entre vários métodos de armazenamento. Estes métodos extrapolam a mera decisão de como serão mantidos os dados em disco, se utilizando simplesmente o próprio sistema de arquivos ou um banco de dados, e chegam até a decisão quanto à distribuição dos dados. A necessidade por esta decisão vem do fato de os membros do grupo envolvidos na cooperação poderem estar distribuídos sobre uma rede, seja ela local, departamental ou ainda uma rede de longa distância como a Internet. Independente do meio ou tecnologia utilizados, apenas a camada de armazenamento deve ficar consciente de sua distribuição, sendo cada instância desta camada localizada sobre uma máquina cliente responsável pela comunicação com as demais instâncias, de acordo com sua necessidade, sem que as outras camadas da arquitetura sequer percebam esta distribuição, como mostra a Fig. 5.4 a seguir.

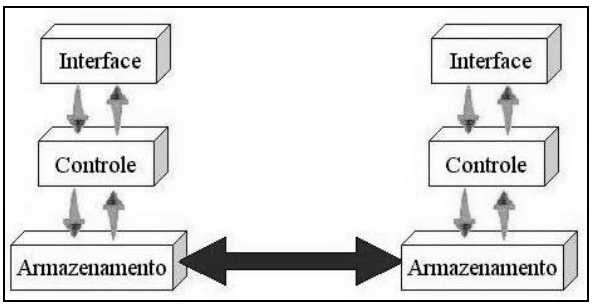


FIGURA 5.4 - Comunicação na camada de armazenamento

Várias são as soluções possíveis para distribuição das informações. As primeiras consistem na distribuição da base de informações entre as máquinas usadas pelos participantes. Cada uma destas máquinas teria sua instância do mecanismo de contextualização mantendo suas próprias informações em uma base. Esta base poderia ser composta tanto por somente informações geradas localmente, quanto por informações geradas em outras máquinas e replicadas localmente. No primeiro caso, a base de informações mantida localmente seria razoavelmente menor, mas em contrapartida a camada de armazenamento em cada máquina fica responsável por transparentemente buscar as informações mantidas pelas demais, o que poderia implicar em um custo de comunicação muito alto, visto que os participantes podem estar distribuídos geograficamente sobre uma rede de longa distância. No segundo caso, cada máquina manteria uma cópia do conjunto completo de informações, fazendo com que a camada de armazenamento não necessite buscar remotamente as informações toda vez que precisasse delas. Entretanto, essa manutenção de toda a base de informações implicaria em um consumo de disco maior, além de incluir o problema do controle de consistência destes dados, uma vez que a camada de armazenamento de cada máquina participante deverá replicar as informações geradas localmente para todas as demais máquinas, gerando, por conseqüência, um grande tráfefo de informações para a manutenção destas bases locais.

Independente de se manter ou não toda a base de informações localmente, a distribuição da camada de armazenamento traz como principal vantagem uma menor vulnerabilidade do mecanismo a possíveis falhas na rede. Essas falhas devem ser sempre consideradas quando se está projetando um *groupware* que opere com participantes em locais diferentes. Quanto mais distantes estiverem estes participantes, maiores serão as

possibilidades de estarem operando através da Internet, a qual não apresenta, até este momento, nenhuma garantia de serviço ou qualidade. Quando se opera sobre Internet, a queda de *links* ou a presença de conexões de baixa velocidade são problemas comuns que podem dificultar o trabalho em grupo, tanto em ambientes síncronos, onde uma conexão lenta, por exemplo, pode dificultar muito a comunicação entre os membros, quanto em ambientes assíncronos, onde uma queda em um *link* pode fazer com que um membro não possa trabalhar sobre informações atualizadas por um longo período de tempo. Por este motivo, ter a possibilidade de uma implementação não tão vulnerável a estes problemas consiste em uma grande vantagem para um *groupware* que pretenda operar sobre a Internet.

Ambas as propostas de distribuição, tanto com a base contendo só infomações locais, quanto contendo todas as informações replicadas, permitem que, mesmo em caso de falhas em *links*, o trabalho em grupo possa seguir, ao custo de uma possível perda de parte da informação mantida nas máquinas falhas. Assim, o uso deste tipo de solução pode minimizar as conseqüências que este tipo de situação pode ter, sendo de grande interesse para sistemas de *groupware* que optem por trabalhar sobre a Internet e que estejam preocupados em evitar que estas conseqüências prejudiquem o trabalho em grupo.

A Fig. 5.5 apresenta de forma geral como o mecanismo de contextualização poderá funcionar com a camada de armazenamento distribuída em caso de falhas no *link* de um participante. A figura ilustra um grupo de três participantes, “A”, “B” e “C”, trabalhando com uma ferramenta de *groupware* sobre a Internet. Quando o *link* do último participante cai, os demais podem seguir seu trabalho normalmente, comunicando-se entre si, via Internet, e até mesmo este último poderá também seguir seus trabalhos, caso a ferramenta de *groupware* o permita, muito embora seu acesso às informações atualizadas, oriundas dos outros participantes, não será possível, da mesma forma que as informações sobre suas atividades permanecerão desconhecidas para seus colegas até o restabelecimento da ligação entre todos.

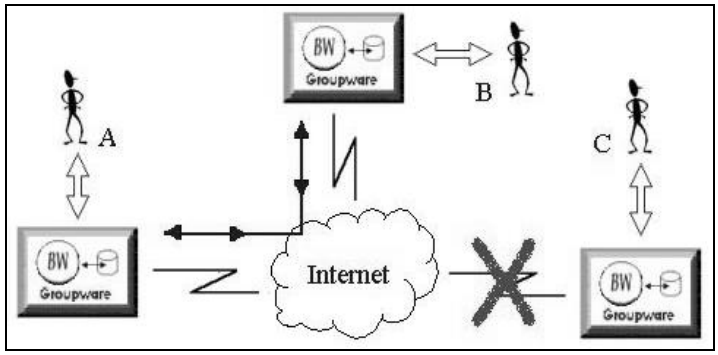


FIGURA 5.5 - Operação com falhas sobre arquitetura distribuída

Embora tenha como grande vantagem a menor vulnerabilidade quanto a falhas na rede, a idéia de distribuir a camada de armazenamento inclui nesta um grau bem maior de complexidade, uma vez que toda a parte de controle de consistência precisa ser tratada dentro desta camada. Existem hoje disponíveis na bibliografia várias opções em algoritmos para distribuição e correta manutenção de informações, dentro desta idéia de distribuí-las entre um conjunto de máquinas. Todavia, estes algoritmos costumam apresentar um grau elevado de complexidade e apresentam limites, fazendo muitas vezes com que a vantagem da menor vulnerabilidade não compense o grande aumento de complexidade da camada de armazenamento embutido no processo.

Como alternativa a estas situações, em que o aumento de complexidade envolvido com a distribuição da camada de armazenamento não compensa suas vantagens, está a utilização do modelo cliente/servidor para esta camada. Dentro deste modelo, todas as informações geradas pelo mecanismo ficam armazenadas em um único servidor central, que é acessado pelas demais máquinas clientes. Estes clientes, que representam as demais instâncias do mecanismo que acompanham os participantes, coletam e enviam a este servidor as informações necessárias para realizar a contextualização de seus participantes.

A Fig. 5.6 apresenta um esquema geral do mecanismo, utilizando o modelo cliente/servidor. Na figura observa-se um servidor central, máquina "A" e dois clientes ligados a este servidor via Internet. Dentro da mesma figura, pode-se observar um terceiro cliente, cuja conexão com a Internet apresenta falhas. Este cliente fica sem total acesso às informações dos demais, já que não pode consultar o servidor. Este seria o máximo nível de falha que este modelo poderia suportar: a falha no cliente. Caso esta mesma falha ocorresse na conexão do servidor junto à Internet, todas as informações de *awareness* mantidas neste servidor ficariam inacessíveis aos seus clientes.

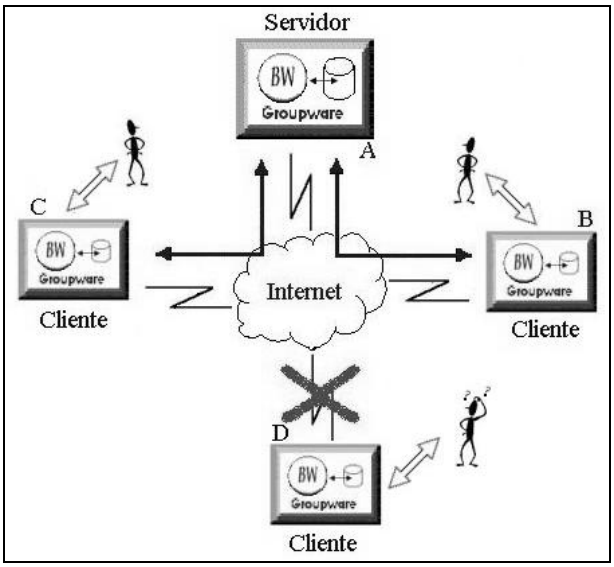


FIGURA 5.6 - Uso do modelo cliente/servidor na camada de armazenamento

Embora esta solução apresente uma grande vulnerabilidade a falhas, especialmente no servidor, sua vantagem reside justamente na simplicidade de implementação. O modelo cliente/servidor é um modelo já bem conhecido e consagrado na computação, sendo utilizado em inúmeros sistemas. Esta popularidade garante um extenso *know-how*, o que facilita bastante a implantação deste tipo de solução. Hoje é bastante comum encontrar-se em diversas linguagens de programação, bibliotecas, *toolkits* e até mesmo componentes, que implementem este modelo de cliente/servidor, fazendo com que sua implementação dentro da camada de armazenamento do mecanismo seja apenas uma questão de adaptação ou uso de elementos como estes.

O uso específico de uma destas formas de distribuição dentro da camada de armazenamento não é uma exigência do mecanismo de contextualização aqui proposto, mas quase uma necessidade na sua inclusão em uma ferramenta de *groupware*. O mecanismo apenas sugere estas opções, com informações locais, replicação ou modelo cliente/servidor, mas não obriga o uso de uma específica. A escolha por uma destas opções depende de como o *groupware* onde o mecanismo será incluído opera: se este *groupware* opera dentro de um modelo cliente/servidor, o uso do mesmo modelo na camada de armazenamento acaba sendo a melhor opção. Já se o *groupware* é totalmente distribuído,

mantendo replicadas suas informações sobre cada um dos *sites* dos participantes, a camada de armazenamento deve acompanhar este modelo. O mecanismo, através do *framework* BW, fornece apenas as bases para estes modelos, mas a implantação propriamente dita de um deles fica por conta do desenvolvedor do *groupware*, responsável pela inclusão do mecanismo neste. As únicas exigências feitas pelo mecanismo são o respeito à estrutura definida no *framework* BW e o armazenamento das informações de acordo com o modelo de dados apresentado na próxima seção. O uso deste modelo garante que todas as informações geradas e necessárias ao mecanismo estarão disponíveis ao mesmo durante o seu funcionamento normal. A próxima seção descreve este modelo de dados adotado.

5.3 Modelagem da Base de Informações

A coleta de informações durante o monitoramento das atividades pressupõe um armazenamento adequado para que estas informações estejam acessíveis posteriormente, no momento da notificação dos participantes quanto às atividades ocorridas. Para tanto, o mecanismo aqui proposto apresenta um modelo de dados que deve ser implementado junto à camada de armazenamento da arquitetura. Este modelo de dados estrutura toda a base de informações utilizada pelo *framework* BW, apresentado no próximo capítulo, procurando evitar redundâncias e facilitar o seu acesso aos dados.

A representação deste modelo de dados apresentada neste capítulo utilizou um Diagrama Entidade-Relacionamento (ER). Um Diagrama ER consiste em um conjunto de objetos básicos, chamados entidades e os relacionamentos entre estes objetos. Uma entidade é um objeto que existe e é distinto dos demais, enquanto que um conjunto-entidade é um conjunto de entidades semelhantes. O mesmo vale para os relacionamentos: um relacionamento é uma associação entre várias entidades e um conjunto-relacionamento é um conjunto de relacionamentos de mesmo tipo. Graficamente, conjunto-entidades são mostrados como retângulos, conjunto-relacionamentos como losangos, e os atributos destes como círculos ligados a estes por retas. [SIL 89],[DAT 91]. A descrição do modelo de dados adotado é então feita nas próximas páginas. Cada segmento do diagrama será apresentado e discutido separadamente para, ao final, apresentar o diagrama completo.

A primeira parte do Diagrama ER modela a questão do registro dos tipos de atividades que serão observadas. Cada tipo de atividade é visto como um tipo de evento diferente e são mantidos em um registro. O trecho do Diagrama ER apresentado na Fig. 5.7 mostra um relacionamento denominado “Preencher” unindo entidades “Groupware” a entidades “Registro”, e esta última sendo associada ao conjunto-entidade “Evento” por um relacionamento 1:N denominado “Descrever”. O primeiro relacionamento representa o processo de registro realizado pelo *groupware* junto ao mecanismo, enquanto o segundo, a relação entre os tipos de atividades registrados pelo *groupware*, que vão ser os tipos de eventos observados, e os eventos gerados, que representam as atividades que realmente ocorreram. Já a entidade “Groupware” representa a ação e a importância do próprio *groupware* junto ao mecanismo e à base de informações.

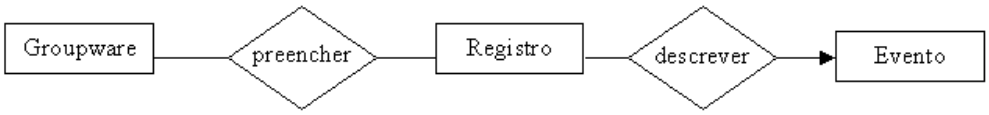


FIGURA 5.7 - Diagrama ER para o registro de eventos

Estes eventos, que representam as atividades ocorridas, são gerados por participantes, que realizaram dentro do *groupware* as atividades correspondentes. Para representar esta relação, o modelo de dados apresenta o relacionamento "Gerar" unindo os conjunto-entidades "Eventos" e "Participantes", como mostra o trecho do Diagrama ER apresentado na Fig. 5.8.

FIGURA 5.8 - Relacionamento entre participantes e eventos

Cada participante dentro do *groupware* pertence a um grupo no qual desempenha um conjunto determinado de papéis. Estes papéis também são determinados pelo grupo, uma vez que grupos distintos poderiam adotar modelos hierárquicos distintos, resultando em papéis diferentes. Toda esta situação é descrita na Fig. 5.9 através de um trecho bastante complexo do Diagrama ER. Neste trecho, diversos relacionamentos unem as entidades "Grupo", "Papel" e "Participante". Há um relacionamento denominado "Determinar" entre "Grupo" e "Papel", representando a relação entre um grupo e os papéis determinados por seu modelo hierárquico. Há também um relacionamento denominado "Possuir" entre "Grupo" e "Participantes" representando que um grupo possui um conjunto limitado de participantes, que individualmente podem participar de vários grupos. E, por fim, há um relacionamento ternário entre estes três conjuntos-entidades, representando o fato de um participante desempenhar um conjunto determinado de papéis em um determinado grupo.

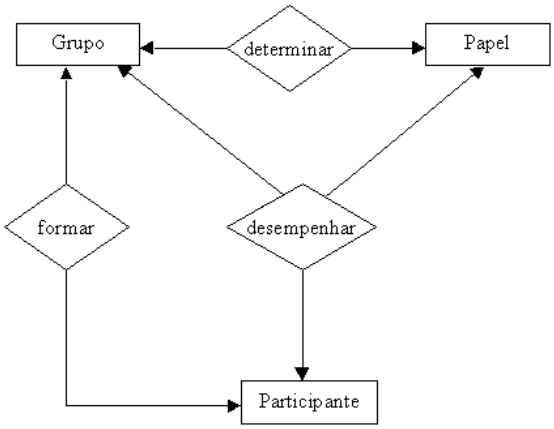


FIGURA 5.9 - Diagrama ER descrevendo as relações entre participantes e papéis em um grupo

Para a filtragem das informações, o mecanismo de contextualização implementa, dentro de seu *framework* BW, o recurso de *profile* sobre os quais cada usuário deposita suas preferências pessoais e do papel desempenhado quanto aos eventos que lhe são interessantes. Estes *profiles* são representados no modelo de dados pelo conjunto-entidade "Awareness Profile" o qual se relaciona aos conjuntos-entidades "Evento", "Papel" e

"Participante pelos conjuntos-relacionamentos "Filtrar", "Interessar" e "Escolher", respectivamente, e liga-se ao próprio relacionamento "Desempenhar" através do relacionamento "Possuir". Com esta estrutura, apresentada no diagrama ER da Fig. 5.10, representa-se os três tipos de *profiles* descritos dentro do mecanismo de contextualização: um para as preferências particulares de cada participante, um para as preferências deste participante, quando ele está desempenhando um papel e outro descrevendo as necessidades de um papel.

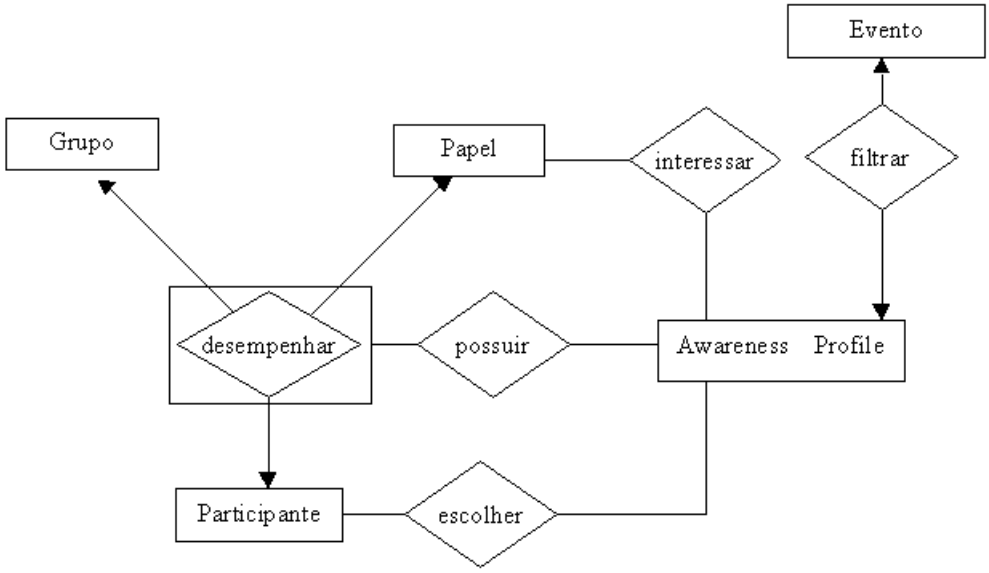


FIGURA 5.10 - Modelo de dados necessário para a contextualização

Por fim, a Fig. 5.11 apresenta o Diagrama ER completo do modelo de dados desenvolvido para o mecanismo de contextualização. Este modelo traz a base de informações completa para o funcionamento deste mecanismo. Esta base é mantida em meio permanente pela camada de armazenamento apresentada na arquitetura, e é sobre todas as suas informações que todo o *framework* BW, descrito no próximo capítulo, opera.

Pode-se perceber no diagrama da Fig. 5.11 a presença da entidade "Setup". Esta é uma entidade especial para fins de implementação. Ela guarda informações importantes para o funcionamento do mecanismo que não se encaixam nas demais entidades. São informações essencialmente de configuração, como um atributo para indicar o endereço local da máquina ("address"), ou o endereço de um possível servidor, no caso de um modelo cliente/servidor, ou ainda um atributo "life time", indicando um tempo, em dias, para a caducidade das informações mantidas na base. Optou-se por um critério de caducidade fixo, com um número de dias, pois o objetivo deste trabalho reside no fornecimento de percepção de eventos no passado, ou seja, de atividades que já ocorreram e já foram concluídas dentro do grupo, e não daquelas que ainda podem sofrer modificações ou que ainda estão ocorrendo. Dessa forma, não são consideradas quaisquer ligações entre eventos, mas apenas um valor fixo, o "life time", como critério de caducidade. Este atributo, "life time", é determinado pelo grupo, de acordo com suas necessidades. Qualquer substituição deste critério por um outro, como relações temporais entre eventos, bem como o interesse por eventos no passado contínuo, são apenas considerados possíveis extensões, trabalhos futuros e não serão tratados aqui.

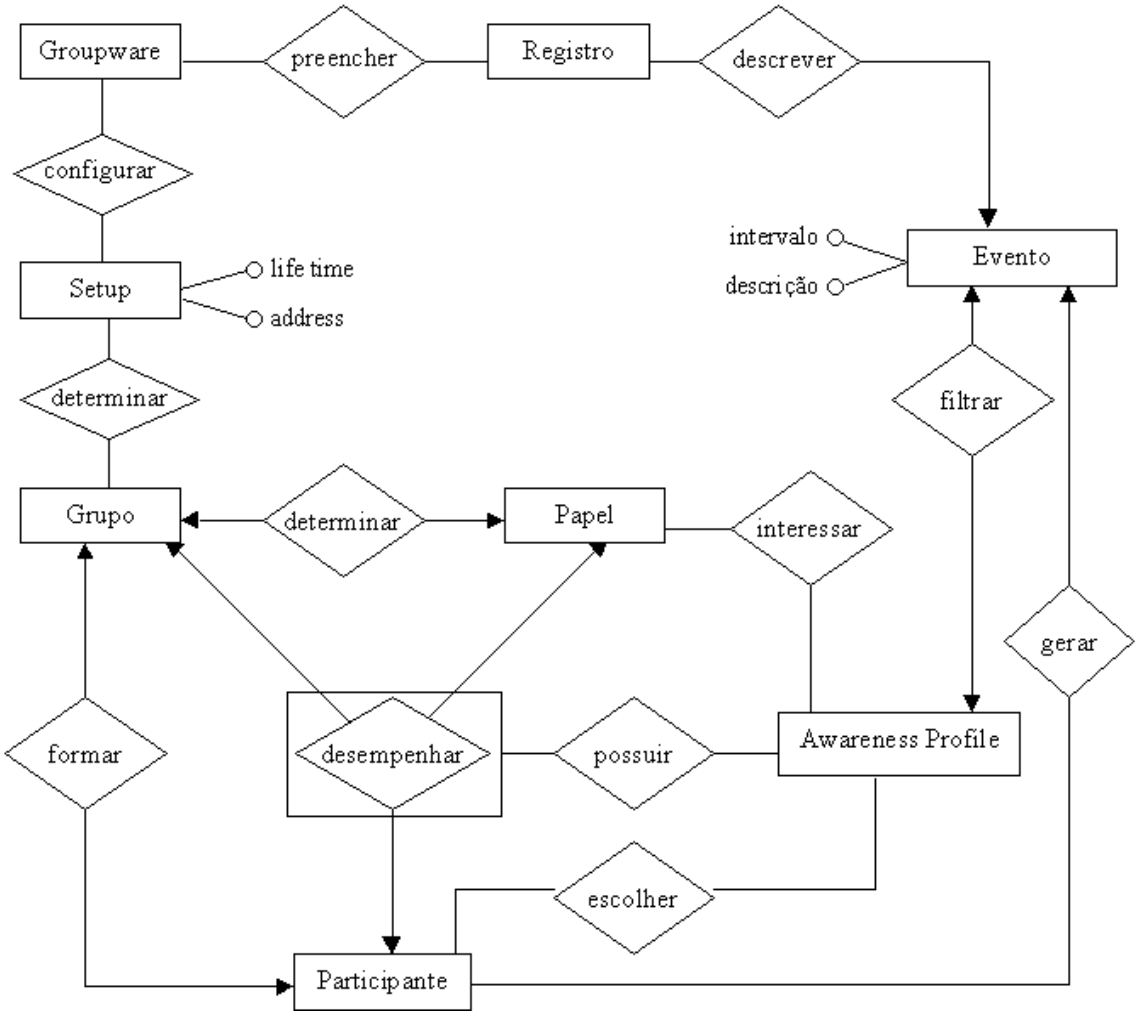


FIGURA 5.11 - Diagrama ER completo do modelo de dados adotado

Como se pode observar pelas figuras 5.9 a 5.11, a estrutura que se formou entre os conjuntos-entidades “Grupo”, “Participantes” e “Papéis” é bastante complexa. Parte desta complexidade se deve ao fato do modelo de dados apresentado pelo Diagrama ER da Fig. 5.11 assumir que o mecanismo de contextualização poderá estar lidando em um mesmo *groupware* com grupos distintos, com estruturas hierárquicas distintas. Isto fornece um grau de flexibilidade maior ao mecanismo, mas embute um grau ainda maior de complexidade a sua implementação. Esta complexidade pode chegar ao ponto de ser desnecessária, uma vez que, na maioria dos casos, as ferramentas de *groupware* impõem um modelo hierárquico, com uma estrutura bem definida para o grupo, visando o tipo de equipe e trabalho que este deseja suportar. Dessa forma, não há, na maioria dos casos, a necessidade de manter as informações sobre mais de uma estrutura hierárquica dentro do mecanismo. Além disso, muitas ferramentas de *groupware* também assumem a presença de um único grupo de usuários, organizados de acordo com o modelo hierárquico utilizado, fazendo com que a definição de mais grupo possa ser desnecessária. Como conseqüência destas situações, o mecanismo, embora apresente em seu modelo teórico a possibilidade de mais de um grupo e modelo hierárquico, trabalhará essencialmente assumindo um único grupo e modelo, definidos pelo *groupware*, com a finalidade de simplificar seu funcionamento.

5.3.1 Atributos de um Evento

Cada entidade "Evento" possui uma série de atributos que lhe caracteriza. Por exemplo, os atributos de descrição e os atributos temporais, como o atributo "intervalo", que descreve o momento de sua ocorrência no tempo. Esta descrição foi feita, baseando-se especialmente em Allen [ALL 83]. Neste trabalho, o autor apresenta uma representação temporal, baseada em intervalos. Cada intervalo "t" é representado por dois pontos no tempo, considerados pelo autor como intervalos muito pequenos os quais determinam o limite superior ("t+") e inferior ("t-") do intervalo. Com base nesta representação, o autor define treze relações possíveis entre dois intervalos (<, >, =, o, oi, m, mi, d, di, s, si, f, fi), apresentadas na Tab. 5.1 a seguir.

TABELA 5.1 - Relações possíveis entre dois intervalos *t* e *s* (baseado em [ALL 83])

Relação	**Símbolo**	**Relação entre os limites**	**Símbolo Inverso**	**Desenho Exemplo**
t before s	t < s	t+ < s-	>	TTTSSS
t equals s	t = s	(t- = s-) ∧ (t+ = s+)	=	TTT SSS
t overlaps s	t ***o*** s	(t- < s-) ∧ (t+ > s-) ∧ (t+ < s+)	***oi***	TTT SSS
t meets s	t ***m*** s	(t+ = s-)	***mi***	TTTSSS
t during s	t ***d*** s	[(t- > s-) ∧ (t+ ≤ s+)] ∨ [(t- ≥ s-) ∧ (t+ < s+)]	***di***	TTT SSSSSS
t starts s	t ***s*** s	(t- = s-) ∧ (t+ < s+)	***si***	TTT SSSSS
t finishes s	t ***f*** s	(t- > s-) ∧ (t+ = s+)	***fi***	TTT SSSSS

O atributo "intervalo" representa um intervalo como o definido por Allen, permitindo não só a representação de atividades com uma certa duração, mas também pontuais, cujos limites superior e inferior do intervalo são iguais, bem como o uso das treze operações apresentadas na Tab. 5.1 para relacionar os eventos entre si. Estes relacionamentos entre os eventos representam, de forma direta, os relacionamentos entre as atividades por eles simbolizadas, fornecendo ao mecanismo um maior poder de expressão que pode ser útil na contextualização dos participantes.

Convém ainda ressaltar que existem outras representações temporais, tais como as por espaço de estados, por linha de tempo e baseadas em Redes de Petri [ALL 83], [WAH 94], [LEI 99]. Entretanto, optou-se por este modelo baseado em intervalos, com as treze operações definidas em Allen [ALL 83], visto que este apresentou o poder de expressão desejado para este trabalho, sem perder em termos de simplicidade.

5.4 Considerações Finais

Este capítulo discutiu em detalhes a arquitetura adotada pelo mecanismo de contextualização aqui proposto. Esta arquitetura divide-se em três camadas bem distintas as quais separam os dados gerados e utilizados da sua apresentação gráfica e do processamento realizado sobre os mesmos. Esta arquitetura vem reforçar a idéia de um mecanismo flexível, à medida que divide a sua complexidade entre as camadas e permite o projeto em separado de cada uma destas camadas. Embora ainda deixe em aberto alguns aspectos, como a forma de distribuição da camada de armazenamento, esta arquitetura exige a utilização de um modelo de dados bem definido, apresentado na Fig. 5.11. Com isso garante-se que o *framework* BW terá a sua disposição um conjunto bem determinado de informações, representados pelas entidades incluídas no modelo de dados. Sem este modelo bem definido e claro, dificilmente se conseguiria chegar a um conjunto de classes que pudesse ser flexível e ainda assim fornecer a percepção sobre eventos no passado. O próximo capítulo parte, então, desta arquitetura e deste modelo de dados discutidos e apresenta o que vem a ser o âmago do mecanismo aqui proposto, o *framework* BW.

6 *Framework* BW

O capítulo anterior apresentou em detalhes a arquitetura e o modelo de dados adotados pelo mecanismo de contextualização proposto neste trabalho. Sobre esta arquitetura encontra-se o *framework* BW, o qual implementa de fato o mecanismo de suporte à percepção de eventos no passado de forma flexível e reutilizável, como desejado. Este capítulo descreve em detalhes este *framework*, apresentando inicialmente a relação entre os eventos utilizados pelo *framework* e as atividades monitoradas, partindo da idéia de *event-based awareness*, seguida pela organização do mesmo, com sua divisão em pacotes, e concluindo com o guia para a produção da interface com o usuário, necessária para a implementação do *framework* junto a um *groupware*.

6.1 Eventos e o Ciclo de Registro – Ocorrência - Notificação

Conforme já foi discutido no Capítulo 4, este trabalho visa a desenvolver um mecanismo para o suporte flexível à percepção de eventos no passado, para que o mesmo pudesse ser adaptado e incluído em mais de uma ferramenta de *groupware* por seus desenvolvedores. O objetivo nesta busca por flexibilidade reside na possibilidade de reaproveitamento e aprimoramento das ferramentas de *groupware* já existentes, de modo que os desenvolvedores destas ferramentas possam utilizar-se deste trabalho para suprir seu *groupware* com o suporte necessário à percepção de eventos no passado. A maior vantagem desta abordagem está no seu custo, pois a adaptação de um mecanismo de suporte à percepção a uma ferramenta de *groupware* seria significativamente menos custosa do que a construção de uma nova ferramenta com tal suporte, ou do que o projeto e desenvolvimento de um suporte específico para tal ferramenta. O reaproveitamento do projeto, do desenvolvimento e do próprio funcionamento do mecanismo de contextualização aqui proposto permite esta redução de custo à medida que a inclusão do suporte à percepção parte de um mecanismo pronto, necessitando somente sua integração ao *groupware*.

Desta maneira, ao se priorizar a flexibilidade, construiu-se um mecanismo de suporte à percepção cuja essência de seu funcionamento é fechada ao *groupware*, podendo até mesmo ser tratada como uma espécie de “caixa preta” por este. A Fig. 6.1 a seguir esquematiza este relacionamento, apresentando o usuário interagindo com o sistema e o mecanismo de contextualização colocado separado do próprio *groupware*.

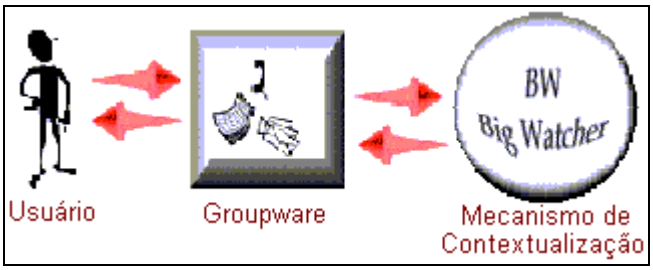


FIGURA 6.1 - Mecanismo de contextualização separado do *groupware*

Esta flexibilidade, que permite ao mecanismo aqui proposto ser esta espécie de "caixa preta", está calçada no desenvolvimento de um *framework* que implemente este suporte e seja capaz de ser adaptado e incluído em ferramentas de *groupware* diversas, baseado na utilização da idéia de *event-based awareness*.

Um *framework* pode ser visto como um conjunto de classes cooperantes que compõe um projeto reutilizável para uma classe específica de *software*. Ele captura as decisões de projeto comuns ao seu domínio de aplicação, enfatizando a reutilização do projeto. O *framework* dita a estrutura da aplicação, mas pode ser adaptado para uma aplicação em particular, principalmente através da criação de subclasses específicas de classes abstratas presentes no *framework*. Assim, ao se usar um *framework*, reutiliza-se seu corpo principal e escreve-se em essência o código que o chama [GAM 94]. O *framework* desenvolvido neste trabalho, intitulado BW (*Big Watcher*), atua sobre a classe de aplicações de *groupware*, capturando importantes decisões de projeto comuns ao suporte a *awareness* de eventos no passado. Com isso, o fornecimento desta percepção é estruturado, de modo que estas ferramentas de *groupware* podem fazer uso deste *framework* para fornecer a seus usuários a percepção destes eventos, tão necessária para a melhor realização do trabalho em grupo.

Este *framework* BW faz uso da idéia de *event-based awareness*, a qual prevê a utilização de eventos como a base para o mecanismo de *awareness*, da seguinte forma: todas as atividades monitoradas são vistas como eventos e, destes eventos ocorridos, é feita a notificação para o usuário, fornecendo ao mesmo a informação do que ocorreu dentro do grupo, formando o contexto das atividades desenvolvidas e, portanto, a percepção do que foi feito. Com isso, *groupware* e *framework* BW comunicam-se somente através destes eventos, os quais carregam as informações sobre tudo o que ocorreu dentro do *groupware* e, por conseqüência, dentro do trabalho em grupo, para dentro do *framework*, que transforma tudo isto em informações para o suporte a *awareness*. Desta forma, o *framework* BW trabalha sobre estes eventos, que representam as atividades do grupo para as quais se deseja fornecer a informação de percepção, em três etapas principais, como mostra o Quadro 6.1 abaixo.

❶ Cada tipo de atividade a ser monitorada é vista como um tipo de evento passível de ocorrência dentro do *framework*;
❷ Quando uma destas atividades ocorre e é concluída dentro do *groupware*, uma instância do tipo de evento correspondente é repassada ao *framework*; este trata e armazena adequadamente cada uma destas instâncias, chamadas aqui simplesmente de "eventos";
❸ Quando lhe for solicitado, o *framework* BW recuperará e enviará, via interface com o usuário, o conteúdo dos eventos ocorridos dentro do *groupware* que forem de interesse do usuário.

QUADRO 6.1 - As três etapas para o suporte à percepção

Estas três etapas representam o ciclo de registro-ocorrência-notificação introduzido no Capítulo 4. Na primeira etapa, é feita a identificação dos tipos de atividades que devem ser monitoradas, através do registro dos tipos de eventos. Na segunda etapa, há a ocorrência das atividades e a conseqüente criação dos eventos dos tipos correspondentes às atividades realizadas e, por fim, na terceira etapa, há a notificação do usuário com a apresentação das informações que são de seu interesse. A Fig. 6.2 abaixo esquematiza este ciclo, apresentando sua ordem de ocorrência e todos os elementos envolvidos (usuário, *groupware* e mecanismo).

FIGURA 6.2 - Ciclo de registro - ocorrência - notificação

Observa-se que há uma correspondência direta entre um tipo de atividade que deve ser observada e um tipo de evento que é passível de ocorrer. Como são as informações das instâncias destes tipos de eventos que são fornecidas como *awareness* aos usuários, cria-se uma forte restrição do *framework* aos tipos de atividades para as quais será fornecida a percepção. A identificação destes tipos de eventos que serão monitorados passa a ser ponto chave para chegar à flexibilidade necessária ao *framework* BW, para que ele possa operar sobre vários *groupwares*. Isto porque a escolha dos eventos é dependente do *groupware* em questão, uma vez que estes representam diretamente as atividades sobre as quais será fornecida a percepção. Por

exemplo, em um editor cooperativo que trabalhe com diversos papéis, a troca de um destes papéis dentro do grupo é um evento importante de ser monitorado, pois pode representar uma alteração significativa na estrutura hierárquica do grupo. O mesmo já não ocorre em ferramentas de *groupware* em que este tipo de troca não é permitido. Outro exemplo, ainda considerando um editor cooperativo que permita a cooperação à nivel de parágrafos, a introdução ou a alteração de elementos desta granularidade consiste em um evento importante, diferente de um editor que trabalhe com granularidades maiores, à nível de capítulo, por exemplo, onde a introdução de um parágrafo novo pouco importará, mas a de um novo capítulo sim.

Dada, então, esta dependência entre os tipos de eventos e os tipos de atividades monitoradas, e entre estes últimos e a própria ferramenta de *groupware*, o *framework* BW optou pela idéia do registro dinâmico destes tipos de eventos, que serão monitorados, com a finalidade de mantê-lo flexível a ponto de poder ser utilizado com diversas ferramentas de *groupware*. Dessa forma, cada *groupware* fará junto ao *framework* BW o registro dos tipos de eventos a serem monitorados, concluindo, assim, a primeira etapa do ciclo de registro – ocorrência – notificação discutido anteriormente.

A Fig. 6.3 traz um diagrama esquematizando este registro, com seu funcionamento geral. O diagrama da Fig. 6.3 enfatiza justamente a troca de mensagens entre os objetos do *framework* BW e o próprio *groupware*. Inicialmente, o *groupware*, representado pelo objeto mais à esquerda no diagrama, envia uma mensagem a um objeto de "fachada" que lhe fornece o acesso ao serviços fornecidos pelo *framework* BW, solicitando o registro de um tipo de evento. Este objeto de fachada limita-se a repassar a mensagem ao real responsável por ela, que é o objeto monitor, incluído como parte integrante da camada de controle. Este objeto monitor trata a requisição feita e repassa o pedido de registro para o objeto que mantém estes registros. Posteriormente, o mesmo objeto monitor também irá solicitar a este último objeto a lista completa dos tipos de eventos registrados, a fim de solicitar a camada de armazenamento, cujos serviços se encontram acessíveis através do objeto mais à direita na Fig. 6.3, a armazenagem destas informações em meio permanente.

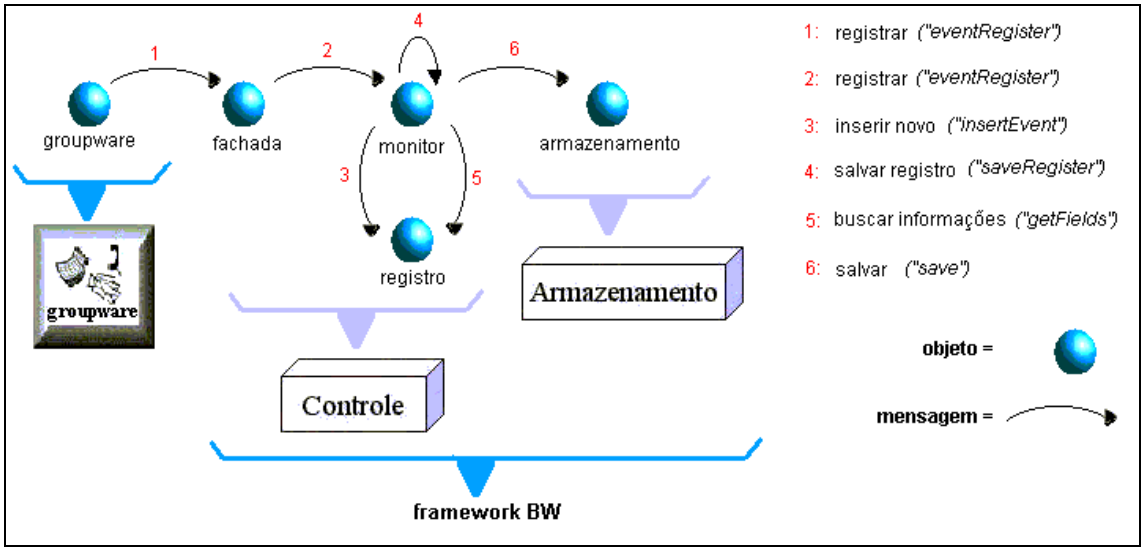


FIGURA 6.3 - Diagrama de seqüência para a operação de registro

Desta maneira, o *groupware* pode dinamicamente personalizar o *framework* BW, fazendo o registro dos tipos de eventos e, portanto, de atividades que serão monitoradas. E graças à utilização de uma classe de fachada, o *groupware* não tem conhecimento

sobre o que está ocorrendo dentro do *framework*, fazendo com que este último se comporte como a "caixa preta" desejada para este trabalho.

Retomando o Quadro 6.1, após realizado o registro dos tipos de eventos/atividades a serem monitorados, a próxima etapa refere-se à ocorrência dentro do *groupware* das atividades relacionadas aos tipos de eventos registrados. Observa-se que somente para as atividades já ocorridas e concluídas dentro do *groupware* são geradas as instâncias dos tipos de eventos correspondentes. Esta restrição faz com que o suporte à percepção fornecido pelo *framework* limite-se à categoria de "eventos no passado" da questão "quando", segundo a classificação apresentada no Capítulo 3. Esta é uma restrição essencialmente de objetivo, pois este trabalho se propôs desenvolver um mecanismo para o suporte à percepção somente de "eventos no passado". Logo, somente atividades que já foram concluídas são de interesse deste mecanismo e, portanto, somente eventos para estas atividades concluídas deverão ser gerados.

Outro ponto interessante a se observar nesta etapa é onde as instâncias dos tipos de eventos são geradas. Este trabalho assume que todas as atividades cooperativas são realizadas dentro do *groupware*. Logo, somente este último tem o conhecimento completo das atividades realizadas e somente ele poderá fornecer todas as informações necessárias para o preenchimento dos eventos correspondentes às atividades realizadas. Assim sendo, quem produzirá as instâncias dos tipos de eventos e as alimentará com as informações das atividades correspondentes será o próprio *groupware*, pois somente ele tem o conhecimento necessário sobre estas atividades já concluídas.

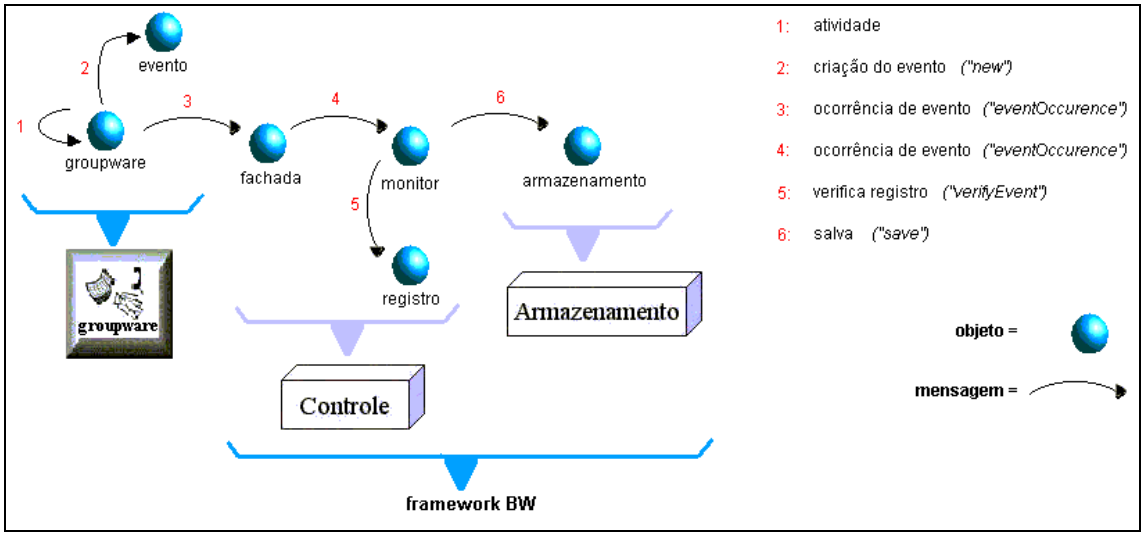


FIGURA 6.4 - Diagrama de seqüência para a ocorrência de um evento

Dessa forma, tão logo é concluída uma atividade, o *groupware* já poderá gerar o evento correspondente e enviá-lo ao *framework* BW que poderá, então, tratá-lo e armazená-lo adequadamente. A Fig. 6.4 traz um diagrama que representa este processo de ocorrência de uma atividade. Inicialmente, há a ocorrência da atividade, colocada sobre o objeto que representa o *groupware* à esquerda na figura. Após concluída a atividade, um evento representando esta atividade (objeto "evento") é criado e passado por meio de uma mensagem para o *framework* BW, através da chamada ao mesmo objeto de "fachada" introduzido na Fig. 6.3. Este objeto dispara internamente ao *framework* o processamento desta ocorrência, enviando uma mensagem ao objeto monitor, responsável por controlar o monitoramento das atividades, o qual verifica se o

evento passado pertence a um dos tipos registrados e somente então ordena o armazenamento deste evento junto à camada de armazenamento, representada pelo objeto mais à direita do diagrama.

Esta segunda etapa do Quadro 6.1, resumida na Fig. 6.4, segue ocorrendo continuamente à medida que o trabalho em grupo prossegue. Entretanto, assim que um participante ingressa neste ambiente ou solicita as informações de percepção, a terceira etapa do Quadro 6.1 tem início. Esta etapa é responsável pela notificação dos usuários, ou seja, pela apresentação das informações de *awareness* ao participante. Esta apresentação ocorre via interface com o usuário, onde é feita a notificação dos eventos ocorridos durante o último período de ausência deste usuário.

Todavia, este período pode ser longo ou de atividades intensas dentro do grupo, gerando um número muito grande de eventos. A apresentação completa de todos estes eventos fatalmente causaria uma sobrecarga de informações sobre o usuário, o que conduz a toda a preocupação com as questões "como" e "quanto" discutidas no Capítulo 3. Nesta ocasião, discutiu-se como o excesso de informações oriundas do mecanismo de suporte à percepção pode prejudicar o trabalho em grupo, à medida que pode interromper demais o usuário ou demandar atenção excessiva de sua parte para a assimilação destas informações. Dessa forma, para evitar que o usuário seja sobrecarregado com estas informações é necessário, no mínimo, um processo de filtragem das mesmas.

Neste trabalho, optou-se por atingir este problema de sobrecarga de informações pela abordagem da questão "como", a qual propõe o uso de mecanismos de filtragem e/ou agrupamento junto à interface com o usuário para evitar que todas (ou nenhuma) informações disponíveis sejam apresentadas. A idéia, neste trabalho, consiste no uso de filtros durante o processo de notificação para limitar o conjunto de eventos que serão apresentados ao usuário. Estes filtros buscam resumir os interesses de cada usuário, procurando apresentar somente aqueles eventos que estiverem de acordo com estes interesses, que sejam considerados úteis e importantes para cada usuário.

Estes filtros são chamados de "*profiles*" e são divididos em três tipos, cada um indicando um conjunto de tipos de eventos que são, de certa forma, interessantes para o usuário. O primeiro destes *profiles*, chamado de "*profile* pessoal", introduz os interesses pessoais do usuário, indicando quais são os tipos de eventos que lhe são particularmente importantes. O segundo tipo de *profile*, chamado "*profile* por papel", traz os interesses do papel que estiver sendo desempenhado pelo usuário, incluindo os tipos de eventos cujas informações são importantes para o melhor desempenho do papel. Cada usuário possui tantos exemplares deste *profile* quantos forem os papéis por ele desempenhados, pois cada papel poderá ter uma necessidade pela percepção de tipos de eventos diferentes, já que estes papéis poderão ter interesse por atividades distintas. O último tipo de *profile*, chamado de "*profile* pessoal por papel", mescla os dois tipos anteriores, representando as preferências particulares do usuário especificamente quando ele estiver desempenhando um determinado papel, e, assim como o "*profile* por papel", cada usuário vai possuir tantos deste *profile* quantos forem os papéis que ele pode desempenhar.

Estes três tipos de *profiles* são combinados durante o processo de notificação, resultando em um conjunto de tipos de eventos que representam todos os interesses do usuário. O primeiro tipo de *profile* vai representar os interesses mais pessoais do usuário, que estarão sempre presentes, independente do papel que ele esteja desempenhando. O segundo vai trazer aqueles tipos de eventos que somente serão

importantes quando um determinado papel está sendo desempenhado pelo participante e encontra-se associado a este papel, não ao participante. O último combina os dois anteriores, trazendo aqueles tipos de eventos que são de interesse do usuário somente quando ele estiver desempenhando um determinado papel. Através da combinação de todos estes *profiles*, espera-se chegar a um conjunto final de tipos de eventos que melhor represente todos os interesses do usuário e que inclua também os tipos de eventos importantes à execução de suas atividades.

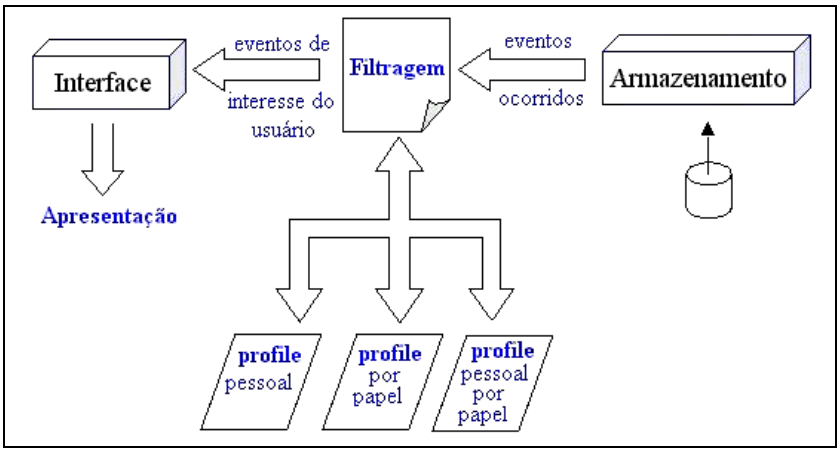


FIGURA 6.5 - Esquema geral para o processamento dos profiles

Este conjunto resultante da combinação dos *profiles* é utilizado para o processo de filtragem, reduzindo o número de eventos apresentados àqueles ocorridos durante o período de ausência do usuário que forem de seu interesse, ou seja, àqueles que satisfaçam o conjunto definido pelos *profiles*. A Fig. 6.5 resume este processo, mostrando os três tipos de *profiles* sendo combinados no processo de filtragem, limitando o conjunto completo de eventos recebidos da camada de armazenamento em somente aqueles eventos de interesse do usuário, os quais são então apresentados via camada de interface.

O diagrama apresentado na Fig. 6.6 a seguir mostra como é feito este processo apresentado na Fig. 6.5 dentro do *framework*. Internamente, a notificação parte da interface com o usuário, pelo ingresso ou por solicitação explícita do usuário, representada no diagrama pelo objeto “gerente interface”. Este objeto repassa a solicitação ao objeto “fachada contexto”, que representa a fachada para o acesso às funções de contextualização na camada de controle. Este objeto, por sua vez, repassa a chamada ao real responsável pelo processamento das requisições de *awareness*, o qual busca os *profiles* junto ao objeto que representa o usuário, incluindo o *profile* por papel, e processa estes *profiles* para só depois solicitar à camada de armazenamento os eventos que se encaixam nestes *profiles*. Estes eventos selecionados são então apresentados ao usuário através dos objetos de interface, representados pelos dois objetos mais à esquerda do diagrama.

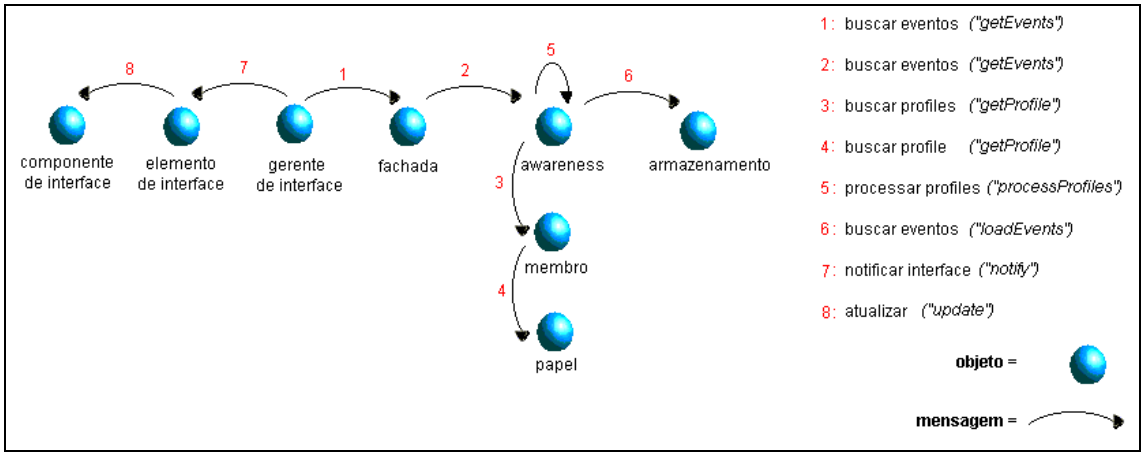


FIGURA 6.6 - Diagrama de seqüência para a notificação

Estas três etapas apresentadas no Quadro 6.1 e discutidas aqui representam o ciclo de registro (Fig. 6.3) – ocorrência (Fig. 6.4) – notificação (Fig. 6.6). Este ciclo é o centro de todo o funcionamento do mecanismo de suporte à percepção implementado pelo *framework* BW. Nas seções seguintes é apresentada em detalhes a estrutura interna deste *framework*, com todas as classes que tornam este ciclo possível.

6.2 Organização do *Framework* BW

A construção do *framework* BW teve como ponto de partida a arquitetura discutida no capítulo anterior. Com isso, todo o *framework* encontra-se dividido, respeitando-se a separação em camadas apresentada na arquitetura, com a finalidade de manter as vantagens de independência entre as partes e divisão da complexidade. Esta divisão foi feita através de "pacotes", os quais encapsulam a definição de cada uma das camadas da arquitetura. Cada um destes pacotes fica responsável por satisfazer as funções de uma camada específica da arquitetura, fornecendo aos demais pacotes os serviços providos por sua camada. Com isso, as classes e as estruturas internas de cada um dos pacotes podem ser projetadas isoladamente, desde que mantidos os serviços ofertados pela camada, obtendo assim, a independência e a divisão da complexidade desejadas.

Dessa forma, o *framework* foi dividido inicialmente em três pacotes denominados "*Storage*", representando a camada de armazenamento da arquitetura, "*Control*", representando a camada de controle, e "*Interface*" para a camada de interface com o usuário. Cada um destes pacotes mantém as mesmas funcionalidades de suas camadas, manutenção, controle e apresentação dos dados, respectivamente, e fornece os mesmos serviços. Todavia, todos os três pacotes apresentam um ponto em comum: as informações que por eles trafegam. Todos os três pacotes lidam com os mesmos tipos de informações, que são essencialmente aquelas geradas pelo monitoramento das atividades concluídas no *groupware* e utilizadas para a contextualização do usuário. Estas informações representam o resultado de todo o ciclo de registro, ocorrência e notificação e formam o que pode ser considerado como o núcleo principal do *framework* BW, uma vez que são estas as informações que serão geradas pelo

groupware, controladas pelo pacote *Control*, armazenadas pelo pacote *Storage* e apresentadas pelo pacote *Interface*.

Logo, com a finalidade de manter a mesma idéia de independência entre as camadas, estas informações não foram incluídas em nenhum dos três pacotes citados acima, pois isto faria com que os demais precisassem ter conhecimento de seu funcionamento interno. Em vez disso, foi criado um quarto pacote, denominado "*Kernel"*, com estas informações utilizadas por todos os demais pacotes e, por este mesmo motivo, consideradas chaves para o *framework*. Este pacote é acessível a todos os demais pacotes, como esquematiza a Fig. 6.7 abaixo, mantendo o seu conteúdo visível aos mesmos, conteúdo este que representa, na forma de classes, o modelo de dados apresentado na Fig. 5.11 do capítulo anterior. Esta visibilidade completa, que de certa forma fere a independência deste pacote, é necessária para o correto funcionamento dos demais, que fazem uso destas informações, além de representar dentro do *framework* a exigência da arquitetura pelo modelo de dados apresentado na seção 5.3.

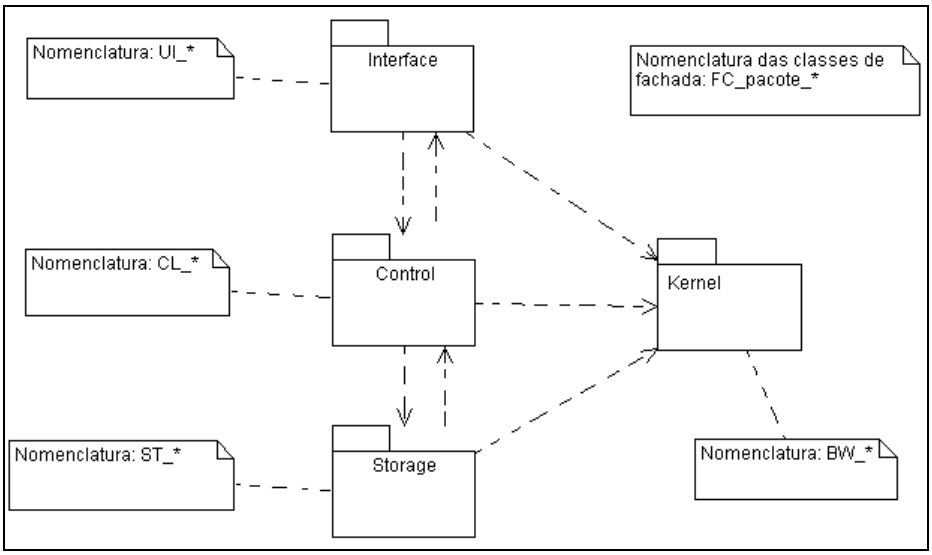


FIGURA 6.7 - Estrutura de pacotes do *framework*

A Fig. 6.7 acima utiliza elementos definidos pela UML - *Unified Modeling Language*. A UML é uma linguagem para modelagem criada pela Rational Corporation, que pode ser usada em todos os processos ao longo do ciclo de desenvolvimento de um sistema [FUR 98]. A Fig 6.7 apresenta a representação gráfica para observações (representadas por uma espécie de nota) ligados por linhas pontilhadas a pacotes da UML (na forma de uma pasta) e as setas pontilhadas representam a existência de dependências entre os pacotes. Pode-se observar que cada pacote representante de uma camada relaciona-se apenas com os pacotes das camadas vizinhas e com o pacote *Kernel*. Posteriormente será mostrado que estas dependências quanto aos pacotes das demais camadas limitam-se a somente algumas poucas classes de fachada. A listagem completa dos símbolos da UML utilizados neste trabalho e seus respectivos significados encontram-se no Anexo 1. Cada um destes pacotes representados na Fig 6.7 serão apresentados nas próximas seções em detalhes, com todas as classes que os compõem e seus relacionamentos.

6.2.1 Pacote Kernel

O pacote *Kernel* introduz dentro do *framework* BW o modelo de dados adotado neste trabalho e apresentado no Capítulo 5, podendo ser visto como uma modelagem UML do mesmo modelo. Trata-se das informações que são produzidas e gerenciadas pelo *framework* como um todo, daí sua importância acentuada dentro deste contexto. Por serem a representação do modelo de dados dentro do *framework*, estas informações devem estar acessíveis a todos os pacotes representantes das três camadas da arquitetura, já que todos estes vão, de alguma forma, manipular estas informações. Deste modo, ao se transportar este modelo de dados para dentro do pacote *Kernel*, garantiu-se que este modelo seria respeitado dentro do *framework* BW, garantindo-se também que estas informações necessárias à percepção de eventos no passado poderão ser geradas e manipuladas corretamente por todo o *framework*.

Como conseqüência, o pacote *Kernel* age como um descritor destes dados gerados e manipulados por todos os demais pacotes. Por este motivo, o conteúdo deste pacote, suas classes e relacionamentos, deverá ser acessível aos demais pacotes, pois o correto funcionamento destes depende do conhecimento das informações que por todos deverão ser manipuladas, ou seja, depende do modelo de dados descrito por este pacote. Este conteúdo do pacote *Kernel*, introduzido no diagrama de classes apresentado na Fig. 6.8 a seguir, deverá estar acessível não só aos demais pacotes do *framework* BW, como também deverá o ser para o próprio *groupware* que estiver fazendo uso do *framework*, pois é através de algumas das classes definidas neste pacote que o mesmo *groupware* poderá transferir suas informações para o mecanismo de contextualização definido pelo próprio *framework*.

Observando-se a Fig. 6.8 a seguir, pode-se perceber que todas as classes no diagrama possuem uma origem em comum, a classe intitulada “BW_Element”. Esta classe representa a noção de elemento básico do *framework* e é a superclasse que dá origem a algumas características importantes presentes em todas as demais classes do pacote: a presença de dois identificadores únicos, um para o objeto e outro para o seu tipo, além da capacidade de ser “clonada” pelo método “clone” e ter seu conteúdo serializado pelo método “getFields”. Estes métodos são úteis para os processos de criação de novas instâncias e de armazenamento e recuperação destas informações em meio permanente, enquanto que os identificadores vão permitir a identificação única de objetos de diversos tipos sem a obrigatoriedade do uso do mecanismo de especialização de diversas classes para a inclusão de novos tipos dentro do trabalho com o *groupware*.

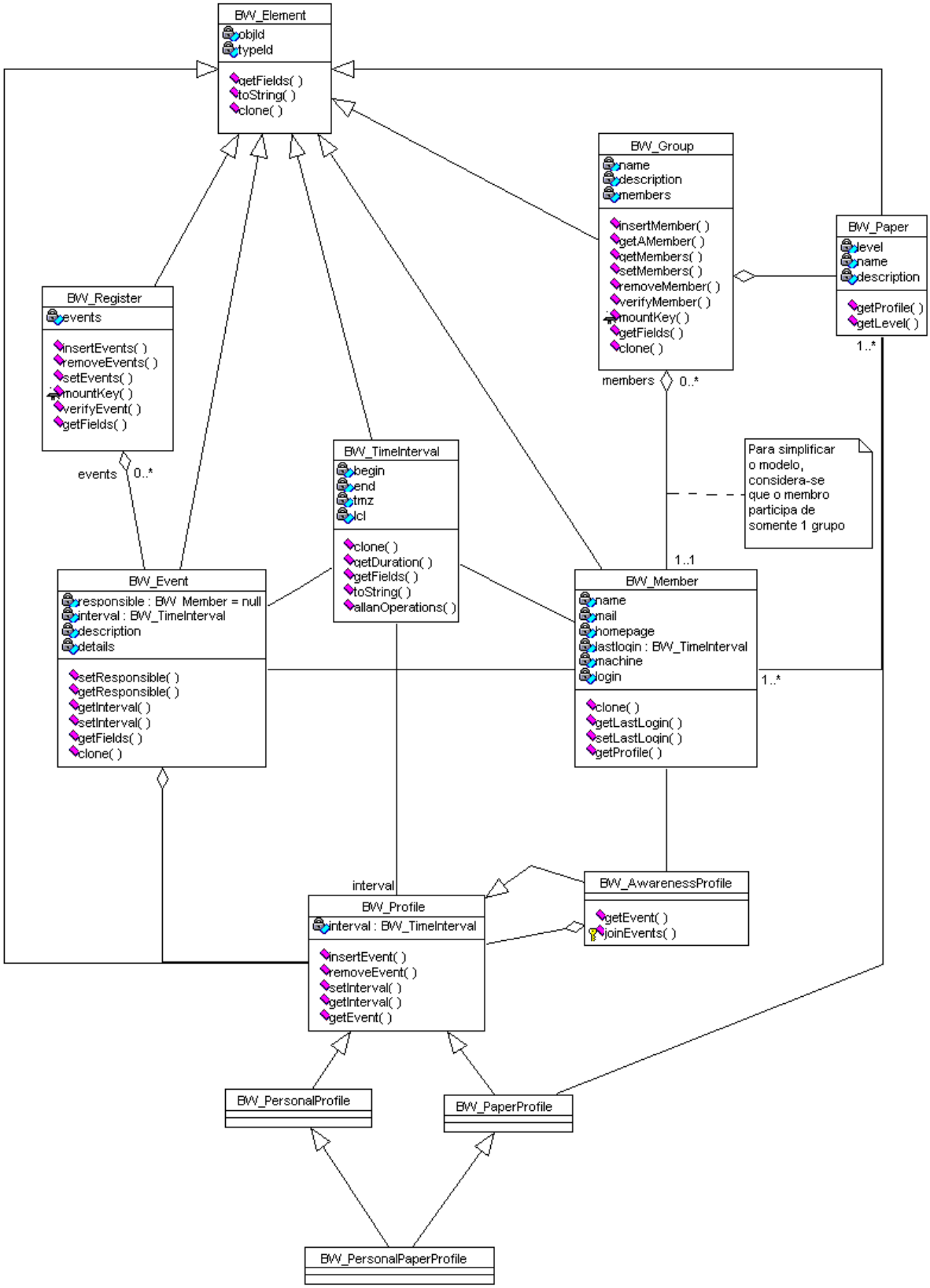


FIGURA 6.8 - Diagrama de classes do pacote *Kernel*

A introdução do método “clone” na superclasse “BW_Element” do pacote *Kernel* trata-se, na verdade, da aplicação do *design pattern Prototype*. Um *pattern* captura a estrutura e a compreensão essenciais de uma família de soluções para um problema que ocorre diversas vezes dentro de um certo contexto e sistema de forças. Tipicamente, um *framework* possui diversos *patterns* em seu interior, favorecendo-o no sentido de que o uso destes *patterns* cria um vocabulário comum para desenvolvedores resolverem estes problemas recorrentes em seus sistemas [APP 99], [DAV 99], [COO 98]. Existem diversas famílias de *patterns* disponíveis na literatura, chamadas de catálogos. O mais famoso destes catálogos talvez seja o definido por Gamma et al. [GAM 94], contendo mais de 20 *design patterns*. Vários destes *design patterns* foram utilizados no desenvolvimento do *framework* BW, e os mais importantes serão apresentados no decorrer deste capítulo, à medida que o próprio BW for sendo discutido.

De modo específico, um *design pattern* sistematicamente nomeia, explica e avalia um projeto recorrente dentro de sistemas orientados a objetos, capturando esta experiência com o objetivo de tornar mais fácil e bem sucedida a reutilização destes projetos. Dentre estes *design patterns*, o denominado *Prototype* especifica tipos de objetos a serem criados usando instâncias protótipo e cria novos objetos, copiando estes protótipos. Seu uso está relacionado a sistemas que devem ser independentes de como seus produtos são criados e a situações onde as classes a serem instanciadas podem ser especificadas em tempo de execução [GAM 94]. A Fig. 6.9, baseada em Gamma et al. [GAM 94], traz a estrutura do *design pattern Prototype*, onde é possível ver a definição de uma superclasse chamada “Prototype”, a qual define o método “clone”, chamado pela classe cliente. Este método é redefinido pelas subclasses, que implementam nela uma operação de cópia, retornando uma nova instância semelhante à original. Já na Fig. 6.10 observa-se um trecho do pacote Kernel, onde a mesma estrutura é utilizada: a superclasse “BW_Element” define o método “clone”, que é implementado nas subclasses, como “BW_Event” e “BW_Member”. O uso deste *pattern* dentro do *framework* BW concentra-se principalmente na carga dinâmica de objetos oriundos da base de dados e na necessidade de independência entre o *framework* BW e as aplicações onde ele estará sendo utilizado, as quais poderão especializar diversas classes, em especial a “BW_Event”, e estas novas classes ainda assim deverão poder ser manipuladas dentro do *framework*.

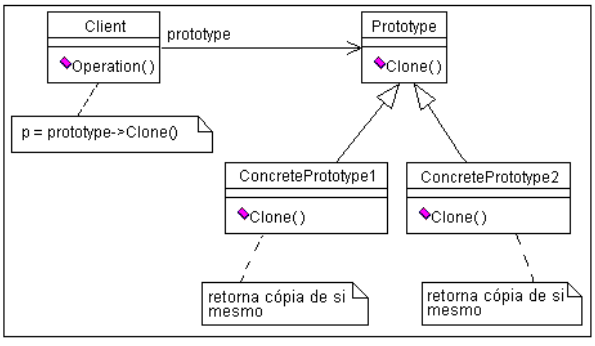


FIGURA 6.9 - *Design pattern Prototype* (baseado em Gamma et. al [GAM 94]).

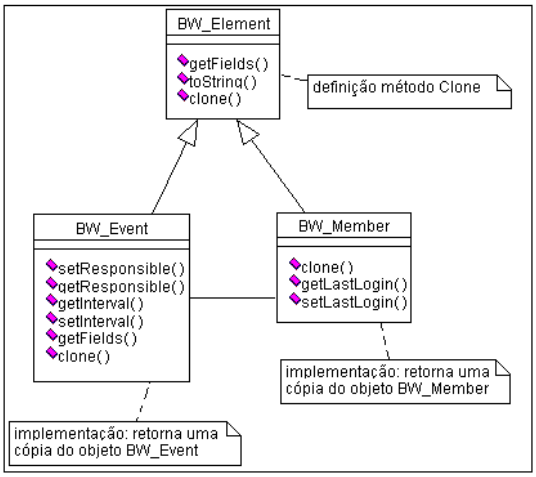


FIGURA 6.10 - Uso do *design pattern Prototype* no pacote Kernel

A classe “BW_Event” é a representante dentro do *framework* BW dos eventos ocorridos, enquanto a classe “BW_Register” mantém o registro destes eventos. Esta classe, “BW_Register” mantém e organiza o registro dos tipos de eventos correspondentes aos tipos de atividades que serão monitoradas pelo *groupware*, mantendo uma instância exemplo, como um protótipo, de cada um destes eventos. As instâncias destes eventos são representadas por objetos da classe “BW_Event”. Cada objeto desta classe vai representar um evento ocorrido, uma atividade de um tipo registrado que já foi concluído dentro do *groupware*, e para descrever o intervalo de tempo em que ocorreu esta atividade é utilizada a classe “BW_TimeInterval” a qual representa a idéia de intervalo de Allen discutida no capítulo anterior. A Fig. 6.11 abaixo mostra todos estes relacionamentos entre estas três classes. A Fig. 6.11 também mostra a possibilidade de especialização da classe “BW_Event” para a inclusão de eventos específicos da aplicação de *groupware*, os quais poderiam ainda ser utilizados dentro do próprio *framework* BW, através do uso de polimorfismo e dos mecanismos de *overriding/overload*, e também poderiam contar com métodos e atributos específicos, que poderiam ser utilizados dentro da aplicação de *groupware* que os definiu.

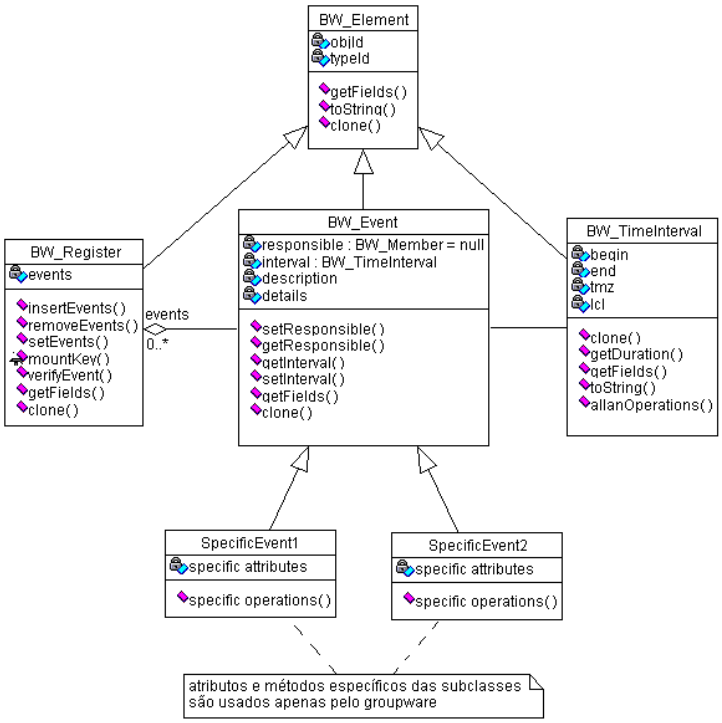


FIGURA 6.11 - Classes para registro e eventos

Conforme pode ser observado ainda na Fig. 6.11, a classe “BW_Register” mantém, além dos métodos herdados da superclasse “BW_Element”, essencialmente métodos para a manipulação do conjunto de tipos de eventos registrados, incluindo métodos para inserção e remoção destes eventos. Através destes métodos, é possível às classes pertencentes ao pacote de controle incluir tipos de eventos a serem observados, durante o processo de registro, cujo diagrama de seqüência foi apresentado na Fig. 6.3, além de poder também remover objetos deste registro e verificar se um determinado tipo de evento está registrado.

Já a classe “BW_Event”, usada para representar as próprias instâncias das atividades concluídas, inclui atributos que descrevem adequadamente estas instâncias, como um atributo para uma rápida descrição do evento (“description”), um responsável pelo evento (“responsible”) e portanto responsável pela atividade realizada dentro do *groupware*, representado por um objeto da classe “BW_Member”, discutida mais adiante, e o intervalo de tempo em que esta atividade foi executada, lembrando sempre que o presente trabalho abrange somente o suporte à percepção de atividades que já foram totalmente concluídas dentro do *groupware*. Este intervalo é descrito segundo Allen [ALL 83], conforme o que foi discutido no capítulo anterior. A classe “BW_TimeInterval” descreve um intervalo de Allen, com sua noção de limites superior e inferior e suas treze operações, resumidas na Fig. 6.11 apenas pelo método “allenOperations”. Além das operações definidas por Allen [ALL 83], que permitem diversas comparações entre dois intervalos, a classe “BW_TimeInterval” possui ainda outros métodos para a manipulação de seus atributos, além dos herdados da superclasse “BW_Element”. O mesmo ocorre também com a classe “BW_Event”, que apresenta diversos métodos para a manipulação de seus atributos, conforme se pode observar na Fig. 6.11, além daqueles herdados de “BW_Event”, decisivos para o armazenamento dos objetos desta classe, através do pacote *Storage*.

A idéia de que possa haver sempre dentro do grupo um membro responsável por uma atividade ou que a tenha executado, foi representada dentro do pacote *Kernel* pelo relacionamento entre as classes “BW_Event” e “BW_Member”, feito através do atributo “responsible” na classe “BW_Event”, o qual indica esta relação entre um membro do grupo e uma atividade por ele concluída. Estas noções de grupo e membro do grupo são representadas dentro do pacote *Kernel* pelas classes “BW_Group” e “BW_Member”, respectivamente. A classe “BW_Group” representa a noção do grupo como um todo, com atributos para sua identificação, como nome (“name”) e descrição (“description”), e métodos para sua manipulação, como ingresso e saída de membros, entre outros, como pode ser visto na Fig. 6.12 mais abaixo. Este grupo é formado por diversos membros, situação esta representada pela relação de agregação entre a classe “BW_Group” e “BW_Member”. Cada objeto da classe “BW_Member” representa um dos membros do grupo, incluindo em seus atributos várias informações a seu respeito, como endereço de correio eletrônico (“mail”), página pessoal (“homepage”) e último período de atividade no sistema (“lastlogin”), além de métodos para a manipulação direta destes atributos.

O grupo também agrega a idéia de papéis, representando a existência de uma hierarquia dentro do grupo. Cada papel é representado por um objeto da classe “BW_Paper”, o qual possui em seus atributos uma identificação básica do papel, incluindo nome (“name”), descrição (“description”) e um indicador de nível hierárquico, para ordenar o papel com relação aos demais, se hierarquicamente superior ou inferior. Já a noção do conjunto de papéis possíveis dentro do grupo é representado pela relação de agregação entre as classes “BW_Group” e “BW_Paper”, considerando-se que um grupo é formado também por um conjunto de papéis e não só por um conjunto de membros, conforme pode ser observado na Fig. 6.12 a seguir, e a noção do conjunto de papéis que um único membro poderá desempenhar dentro do mesmo grupo é representado pelo relacionamento entre “BW_Member” e “BW_Paper”. Além disso, esta noção básica de papel pode ser estendida com a introdução de subclasses de “BW_Paper”, adaptando-se o modelo hierárquico interno ao *framework* àquele adotado pelo próprio *groupware*, como sugerem as classes “SpecificPaper1” e “SpecificPaper2” na Fig. 6.12.

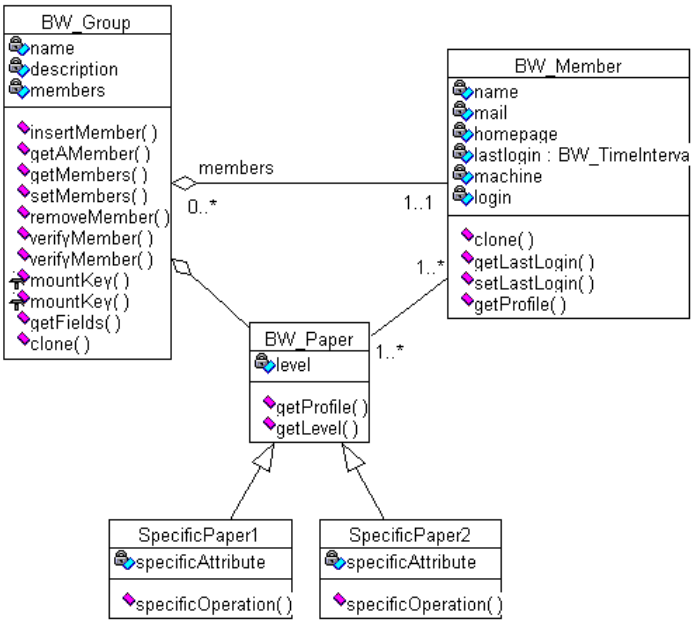


FIGURA 6.12 - Relação entre grupo, membro e papel

Comparando-se a Fig. 6.12 com o modelo de dados apresentado no capítulo anterior, pode-se perceber que foi feita uma severa simplificação desta parte do modelo. Dentro do modelo de dados apresentado na seção 5.3, pode-se claramente observar um relacionamento ternário entre participantes, grupo e papel, além de relacionamentos binários entre grupo e participante, e entre grupo e papel. Estes relacionamentos representam a noção de que um participante poderia ser membro de mais de um grupo, que cada grupo teria associado um conjunto de papéis possíveis, e que um participante seria capaz de desempenhar um subconjunto de papéis em um grupo. Em outras palavras, a noção de um membro desempenhando um papel estaria sempre associada ao grupo onde ele estiver desempenhando este papel. Por exemplo, uma mesma pessoa poderia participar de dois grupos, podendo desempenhar papéis de coordenador e escritor em um, e somente de revisor no outro.

Entretanto, o que se vê no diagrama de classes da Fig. 6.12 são apenas relacionamentos simples entre as classes “BW_Member”, representando os participantes, “BW_Paper”, representando os papéis, e “BW_Group”, representando o grupo. Estes relacionamentos visíveis na Fig. 6.12 apresentam um grupo formado por um conjunto de membros e um conjunto de papéis (agregações entre “BW_Group” e as classes “BW_Member” e “BW_Paper”, respectivamente), e cada membro se relaciona com apenas alguns papéis, em uma associação simples entre “BW_Paper” e “BW_Member”. Ao se observar as cardinalidades destes relacionamentos, percebe-se que cada membro dentro do pacote *Kernel* pode pertencer a somente um grupo (cardinalidade 1..1), e não mais a vários grupos, como o modelo de dados da Fig. 5.11 sugeria.

Esta não é a única simplificação realizada dentro do *framework* BW. Este, além de assumir que cada membro pertence a somente um grupo, presume também que o *groupware* opera com somente um grupo, ou seja, com uma única instância de “BW_Group”. Com isso, resume-se também o modelo hierárquico a um único conjunto

de papéis relacionado a este grupo único, o que faz com que um simples relacionamento entre “BW_Member” e “BW_Paper”, como o presente na Fig. 6.12, seja o suficiente para indicar todo o conjunto possível de papéis que um membro poderá desempenhar.

Estas simplificações tencionam tornar o *framework* mais simples e de implementação mais fácil, com pouca perda de significado, já que as idéias principais de papéis relacionados a um grupo, e membros, pertencentes a este grupo, podendo desempenhar um subconjunto destes papéis, permanecem dentro do modelo de classes. Todavia, caso se desejasse manter o modelo de dados apresentado no capítulo anterior, ao invés de ter optado por estas simplificações, o modelo de classes do pacote *Kernel* não seria muito alterado A Fig. 6.13 traz uma das possíveis alternativas para este caso, ao tentar buscar uma equivalência para o relacionamento ternário presente no modelo de dados da Fig. 5.11 utilizando-se para isso um esquema de “classe associação” representando os papéis, ligando as classes “BW_Group” e “BW_Member”. Uma classe associação é um elemento de modelagem que pode ser vista como uma associação com propriedades de classe ou uma classe com propriedades de associação [FUR 98] e, neste caso específico, estas propriedades podem ser combinadas para demonstrar que a associação entre um membro e um grupo traz sempre consigo os papéis que podem ser desempenhados por este membro.

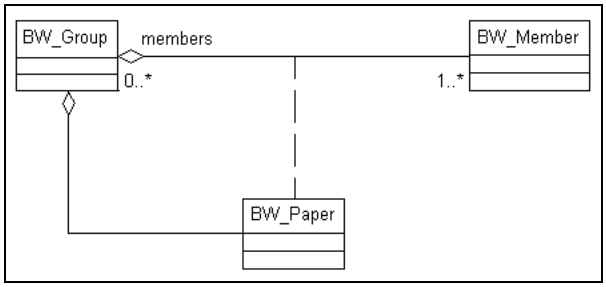


FIGURA 6.13 - Alternativa para a relação entre membros, grupo e papel.

Também relacionada aos participantes e aos papéis (classes “BW_Member” e “BW_Paper”), encontra-se a questão dos *profiles*. Conforme foi discutido no Capítulo 4 e na seção 6.1 deste capítulo, o mecanismo de suporte à percepção proposto neste trabalho utiliza um processo de filtragem de eventos através de *profiles,* com o intuito de notificar ao usuário somente aqueles eventos ocorridos que forem de seu interesse. Estes *profiles* estão representados dentro do *framework* BW pela classe “BW_Profile” e suas subclasses, localizadas também dentro do pacote *Kernel,* as quais podem ser visualizadas pelo trecho do diagrama de classes do pacote apresentado na Fig. 6.14 a seguir.

Através da Fig. 6.14, percebe-se a presença de uma classe base que descreve um *profile* genérico, chamada “BW_Profile”. Esta classe contém a base que descreve todos os *profiles*, incluindo um intervalo de tempo (relacionamento com a classe “BW_TimeInterval”), que descreve o intervalo de interesse do usuário e a agregação dos tipos de eventos nele incluídos, nos mesmos moldes do processo de registro. Esta classe é especializada em *profiles* mais específicos, “BW_PersonalProfile”, “BW_PaperProfile” e “BW_PersonalPaperProfile”, os quais representam os *profiles* “pessoal”, “por papel” e “pessoal por papel”, respectivamente, discutidos na seção 6.1. Todos estes *profiles* são, então, agrupados por um único *profile,* relacionado a cada membro do grupo, o chamado “BW_AwarenessProfile”, que é responsável por fazer a reunião de todos os demais para o processo de filtragem, resultando em um conjunto final de tipos de eventos que representam todos os interesses do usuário.

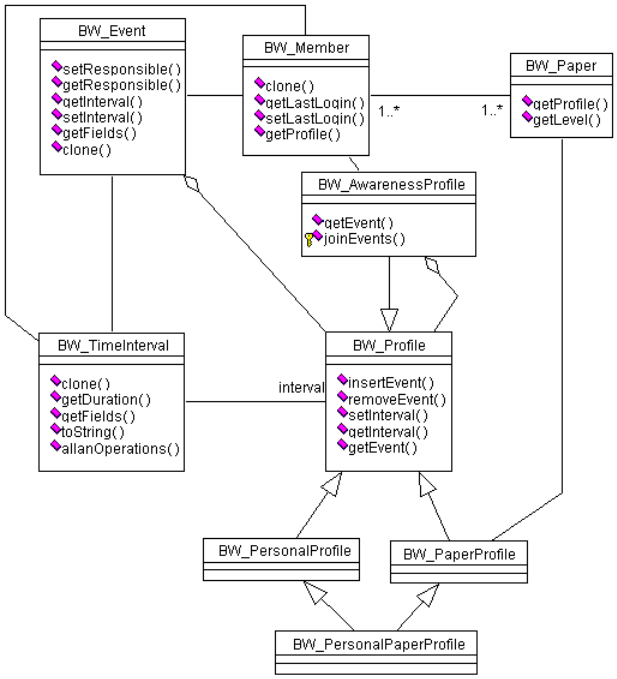


FIGURA 6.14 - Organização dos *profiles* dentro do pacote *Kernel*

Observa-se claramente na Fig. 6.14 uma semelhança entre os relacionamentos das classes “BW_AwarenessProfile” e “BW_Profile” e suas subclasses e o *design pattern Composite* (Fig. 6.15). Este *pattern* busca compor objetos em estruturas de árvore para representar hierarquias de parte-todo ou permitir que clientes tratem objetos individuais ou composições de forma uniforme [GAM 94], como mostra sua estrutura apresentada na Fig 6.15 abaixo. Baseando-se nesta hierarquia de todo-parte proposta pelo *design pattern Composite*, construiu-se o modelo de *profiles*, em detalhe na Fig. 6.16, o qual mantém a mesma idéia das relações de agregação e especialização entre “Component” e “Composite” (Fig. 6.15), representados por “BW_Profile” e “BW_AwarenessProfile” (Fig. 6.16), e a idéia das chamadas sucessivas a um mesmo método em cada um dos componentes dentro do “Composite”, utilizada principalmente pelo método “getEvent” das classes de *profile* presentes na Fig. 6.16, durante o processo de filtragem.

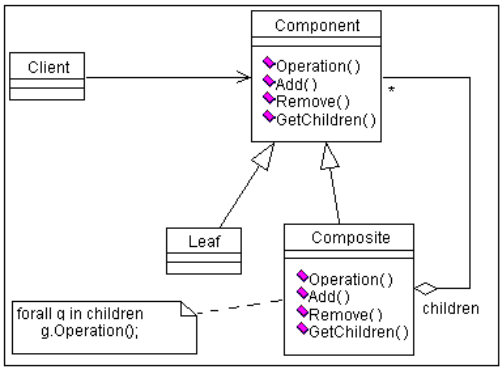


FIGURA 6.15 - *Design pattern Composite* (baseado em Gamma et al [GAM 94])

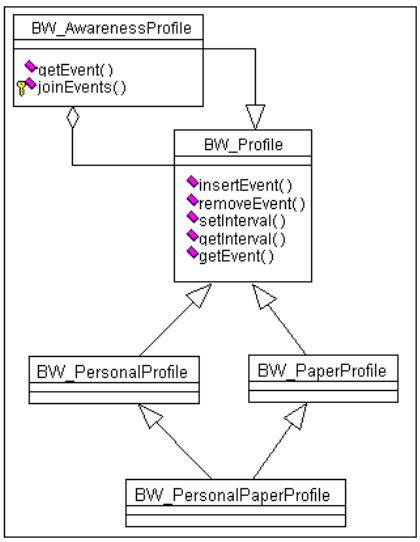


FIGURA 6.16 - Esquema de *profiles* baseado no *pattern Composite*

Todas estas informações presentes no pacote *Kernel* estão disponíveis aos demais pacotes, representantes das três camadas propostas na arquitetura, discutidas no capítulo anterior. Estes outros pacotes vão manipular e gerir as informações descritas no pacote *Kernel* e serão discutidos um a um na seqüência.

6.2.2 Pacote Storage

O pacote *Storage* traz para dentro do *framework* BW as funções da camada de armazenamento. Seu objetivo principal é manter as informações geradas pelas demais camadas em meio permanente. Para tanto, este pacote oferece ao pacote vizinho (pacote *Control*) diversos serviços, os mesmos oferecidos pela camada de armazenamento à camada vizinha. Estes serviços são essencialmente dois: salvar os elementos de informação em meio permanente e resgatá-las. Internamente, o pacote *Storage* também realiza todo o controle deste meio de armazenamento, incluindo inicialização e métodos de acesso a este meio. Com isto, este pacote consegue ocultar dos demais todos os detalhes sobre o meio utilizado e até mesmo qual é exatamente este meio.

A Fig. 6.17 a seguir traz o diagrama de classes completo do pacote *Storage*. Observa-se nesta figura a presença de algumas classes descritas no pacote *Kernel*, como a classe “BW_Element” e “BW_Event”. Estas classes são utilizadas dentro do pacote *Storage* para a manipulação das informações que serão armazenadas e resgatadas do meio físico, demonstrando a idéia do conteúdo do pacote *Kernel* ser aberto aos demais, já que ele traz a descrição dos dados utilizados.

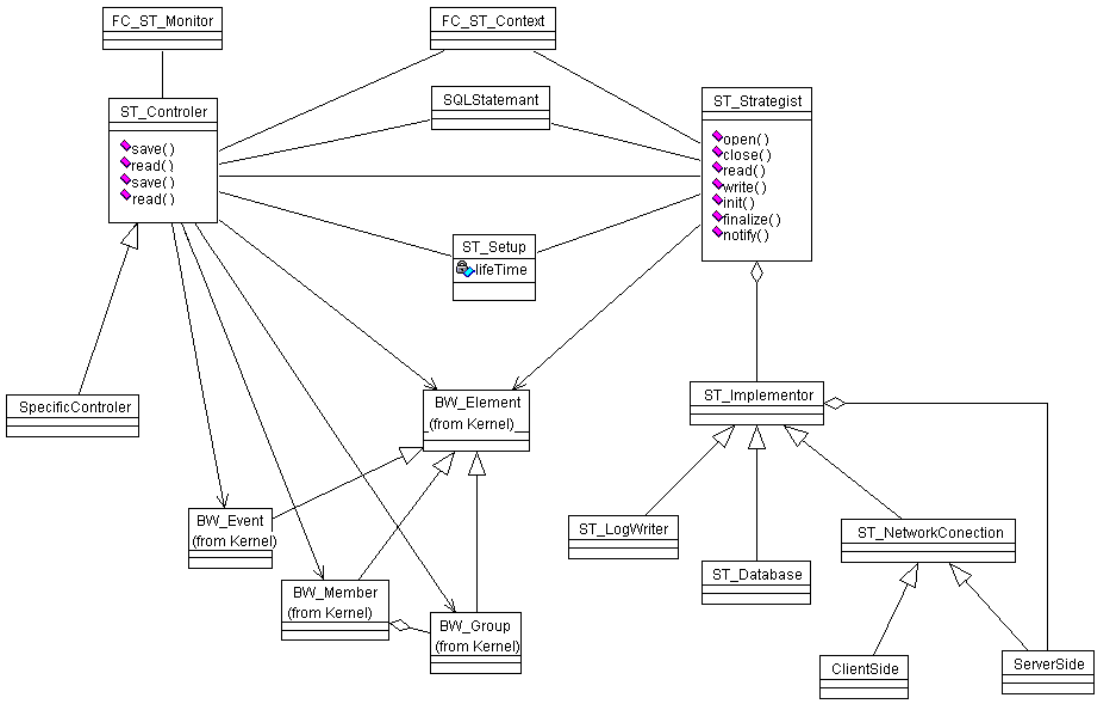


FIGURA 6.17 – Diagrama de classes do pacote *Storage*

Dentro do diagrama apresentado na Fig. 6.17, é possível perceber duas classes principais, a classe “ST_Controler” e “ST_Strategist”. Estas duas classes representam o núcleo do pacote Storage. A primeira, “ST_Controler”, é responsável por tratar todas as requisições de serviços oriundas das camadas vizinhas da arquitetura, enquanto que a segunda, “ST_Strategist”, lida diretamente com o meio de armazenamento.

A classe “ST_Controler” recebe todas as requisições para salvar e resgatar as informações geradas e manipuladas pelas demais camadas da arquitetura, representadas no *framework* BW pelos pacotes *Control* e *Interface*. Estas informações estão descritas pelas classes do pacote *Kernel*, único pacote cujo conteúdo é aberto e conhecido pelo pacote *Storage* e são essencialmente eventos, representados pela classe “BW_Event”, membros, objetos da classe “BW_Member”, e o grupo “BW_Group”. Todos os demais tipos de informação, como papéis (classe “BW_Paper”) e *profiles* (classe “BW_Profile”), podem ser ainda manipulados por polimorfismo, através dos métodos definidos na superclasse “BW_Element” e foram omitidos do diagrama presente na Fig. 6.17 para simplificá-lo.

Todas as requisições enviadas para a classe “ST_Controler” chegam nesta através de duas únicas classes, “FC_ST_Monitor” e “FC_ST_Context”, localizadas na parte superior da Fig. 6.17. Estas são as únicas classes do pacote *Storage* que são visíveis aos demais e representam a fachada externa deste. A primeira delas, “FC_ST_Monitor”, recebe todas as solicitações durante o processo de monitoramento, enquanto a segunda, “FC_ST_Context”, recebe as solicitações durante o processo de contextualização. Dessa forma, o fluxo de comunicação entre o pacote *Storage* e os demais é dividido em dois, de acordo com o sentido das informações: durante o processo de monitoramento, a tendência é que as informações fluam da camada de controle, pacote *Control*, para a camada de armazenamento, pacote *Storage*, à medida que os eventos vão sendo gerados. Já durante o processo de contextualização, a maior parte das informações flui no sentido contrário, do pacote *Storage* para o *Control*, à medida que os usuários forem retornando ao sistema e solicitando as informações de percepção.

O uso destas duas classes por onde passa toda a comunicação com o pacote é, na verdade, a implementação de mais outro *design pattern*, chamado *Facade*. O *pattern Facade* tem por objetivo oferecer uma interface unificada para um conjunto de interfaces em um subsistema, de modo a tornar seu uso mais simples. Sua aplicação ocorre normalmente quando se deseja fornecer uma interface simples para um subsistema complexo, quando se deseja desacoplar um subsistema de clientes e outros subsistemas, e ainda quando se deseja dividir um subsistema em camadas, criando um ponto de entrada para cada nível do subsistema [GAM 94]. A Fig. 6.18 abaixo, baseada em Gamma et al. [GAM 94], mostra a estrutura deste *design pattern*.

Dentro do *framework* BW, o *pattern Facade* foi utilizado para dividir todo o mecanismo em camadas independentes, representadas no *framework* por seus pacotes, de acordo com a arquitetura definida. Através da aplicação do *design pattern Facade*, cada pacote possui sempre duas classes de “fachada”, que são as únicas classes visíveis ao pacote representante da camada superior, e servem justamente para sua comunicação com este outro pacote. Cada pacote mantém ainda contato com somente as classes de fachada do pacote representante de sua camada inferior, e é através destas classes que este pacote faz suas requisições ao seguinte, sem manter qualquer outro conhecimento sobre o conteúdo deste último.

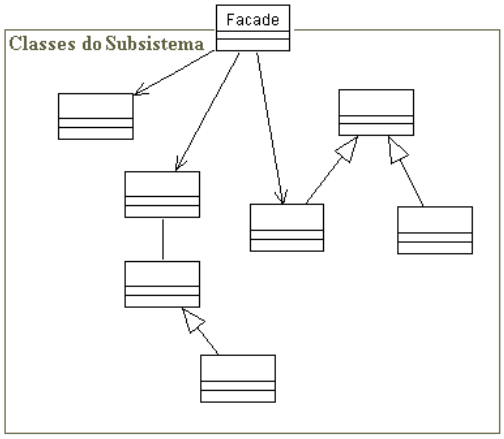


FIGURA 6.18 - Estrutura do *design pattern Facade* (baseado em Gamma et al. [GAM 94])

Especificamente dentro do pacote *Storage* existem as classes de fachada "FC_ST_Monitor" e "FC_ST_Context" que fazem a sua comunicação com o pacote *Control*, o qual representa a camada de controle da arquitetura, colocada imediatamente acima da camada de armazenamento. A Fig. 6.19 abaixo traz a estrutura que as classes "FC_ST_Monitor" e "FC_ST_Context" compõem dentro do pacote *Storage*, a qual pode ser comparada à estrutura básica do *design pattern Façade* apresentada na Fig. 6.18, baseada em Gamma et al. [GAM 94].

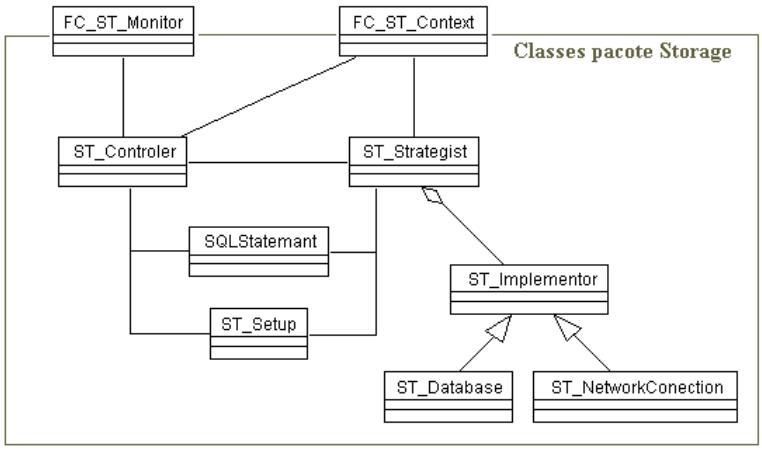


FIGURA 6.19 - Aplicação do *pattern Facade* no pacote *Storage*

Outro *design pattern* utilizado dentro do pacote *Storage* é o *Bridge*, usado junto à classe "ST_Strategist", para controlar o acesso direto ao meio físico que armazena as informações. A classe "ST_Strategist" escolhe o método de acesso ao meio utilizado, o qual é efetivamente implementado pela classe "ST_Implementor" e suas subclasses. Com isso se separa a implementação do acesso ao meio físico, feita por "ST_Implementor" e subclasses, de sua representação dentro do próprio pacote *Storage*, feita por "ST_Strategist", permitindo que método de acesso e o próprio meio de armazenamento possam variar sem que o restante do pacote seja afetado por isso.

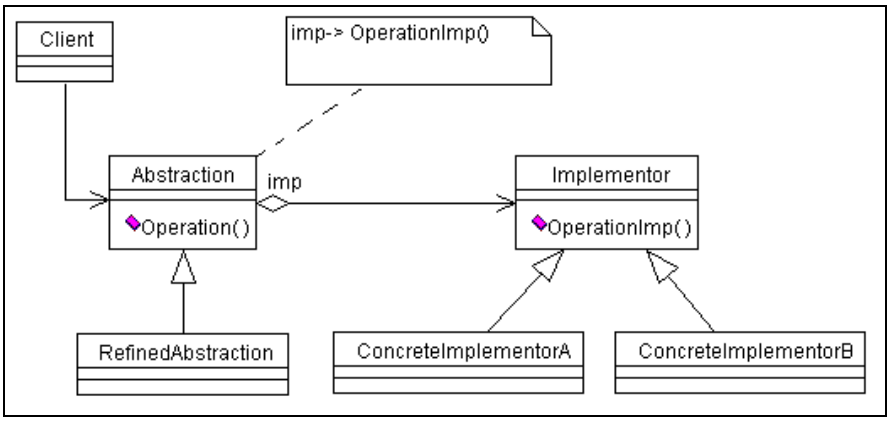


FIGURA 6.20 - *Design pattern Bridge* (baseado em Gamma et al. [GAM 94])

Esta idéia de separação entre a abstração e a implementação real é justamente o objetivo do *design pattern Bridge*. Segundo Gamma et al. [GAM 94], a intenção do *Brigde* é desacoplar uma abstração de sua implementação, de modo que ambas possam variar independentemente. Suas aplicações incluem sistemas onde tanto a abstração quanto a implementação podem ser especializadas, sistemas onde alterações na implementação não podem impactar nos clientes, e sistemas onde a implementação pode ser escolhida em tempo de execução. Sua estrutura é apresentada na Fig. 6.20 acima e pode ser comparada com sua aplicação dentro do pacote *Storage*, apresentada na Fig. 6.21. Nesta última, é possível se observar a classe "ST_Strategist" agindo como abstração, podendo selecionar a implementação, objeto da classe "ST_Implementor", adequado à situação. Em ambas as classes, é possível o uso do recurso de especialização, a fim de melhor adaptar o *framework* BW a sua situação de trabalho, como indica a Fig. 6.21.

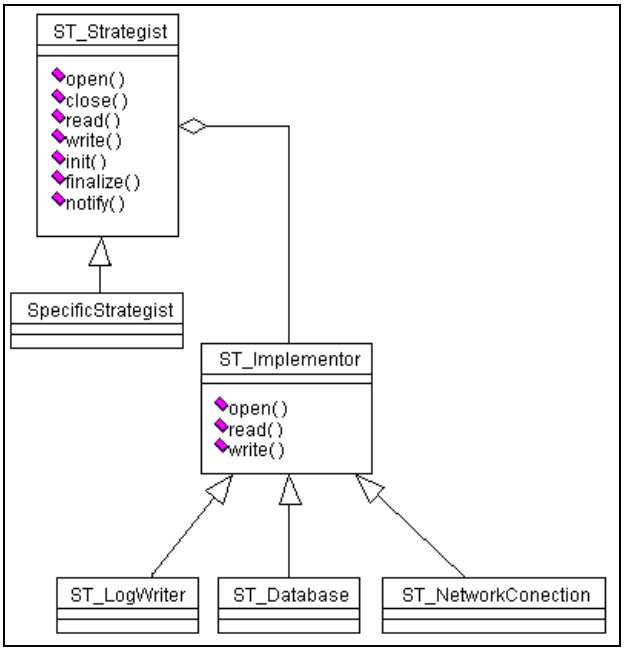


FIGURA 6.21 - Aplicação do *pattern Bridge*

Na Fig. 6.21 é ainda possível se observar três possibilidades de implementação, através das subclasses de “ST_LogWriter”, “ST_Database” e “ST_NetworkConnection”. A primeira, “ST_LogWriter”, implementa um armazenamento utilizando-se do sistema de arquivos puro, através de um esquema simples de “logs” de eventos e informações, enquanto a segunda, “ST_Database”, realmente implementa a idéia de acesso a um banco de dados. A terceira, por sua vez, lida com conexões de rede, permitindo a implementação do modelo distribuído da camada de armazenamento, discutido no Capítulo 5. O diagrama de classes presente na Fig. 6.21 sugere o uso do modelo cliente-servidor, com subclasses de “ST_NetworkConnection” para cada situação, e a presença de um segundo uso do *design attern Bridge* no lado servidor, para descrever a implementação utilizada por este para manter todas as informações.

Retornando a Fig. 6.17, pode-se observar ainda a presença de outras duas classes ainda não discutidas, a classe “SQLStatement” e a classe “ST_Setup”. Estas são duas classes mais voltadas para a implementação prática do *framework* BW. A primeira delas fornece uma representação de sentenças SQL, utilizadas pela classe “ST_Controler” para melhor descrever as requisições feitas à classe “ST_Strategist”, enquanto a segunda, “ST_Setup”, é a representação da entidade “Setup”, presente no modelo de dados discutido no Capítulo 5 (ver Fig. 5.11). Esta entidade traz alguns detalhes importantes para a implantação da camada de armazenamento, incluindo o atributo “lifeTime”, que age como critério de caducidade para as informações mantidas em meio permanente.

6.2.3 Pacote Control

O pacote *Control* descreve dentro do *framework* a camada de controle da arquitetura. Seu objetivo é gerenciar as informações resultantes do monitoramento e organizá-las para a posterior contextualização de cada usuário. Trata-se, portanto, do pacote “pensante” do *framework*, já que ele realiza grande parte do processamento do mecanismo de suporte à percepção como um todo. É este pacote que recebe o registro e os próprios eventos ocorridos do *groupware*, trata-os e os envia para armazenamento junto ao pacote *Storage*, para posteriormente solicitá-los novamente, processá-los e enviar o conjunto de eventos filtrado resultante para o pacote *Interface*, que o apresentará ao usuário final. A Fig. 6.22 a seguir esquematiza bem todo este ciclo de funções. Na figura, é possível perceber como a camada de controle, representada dentro do *framework* pelo pacote *Control*, é a que mais serviços fornece as demais, executando tanto o registro dos tipos de eventos, quanto o processamento das ocorrências destes eventos (atividades 2 e 5 na Fig. 6.22), além da filtragem dos eventos ocorridos para a notificação do usuário (atividade 7).

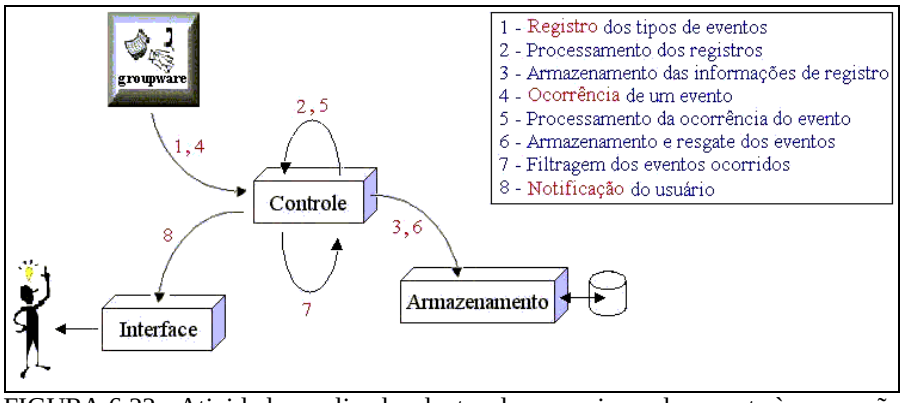


FIGURA 6.22 - Atividades realizadas dentro do mecanismo de suporte à percepção

Para fornecer estes serviços apresentados na Fig. 6.22, o pacote *Control* conta com o modelo de classes apresentado na Fig. 6.23 abaixo. Observando-se a Fig. 6.23, é possível perceber duas classes principais no diagrama: a classe "CL_Monitor" e a classe "CL_Awareness". Estas são as classes centrais deste pacote, responsáveis por todo o processamento das requisições resultantes do monitoramento, classe "CL_Monitor", e da contextualização, "CL_Awareness".

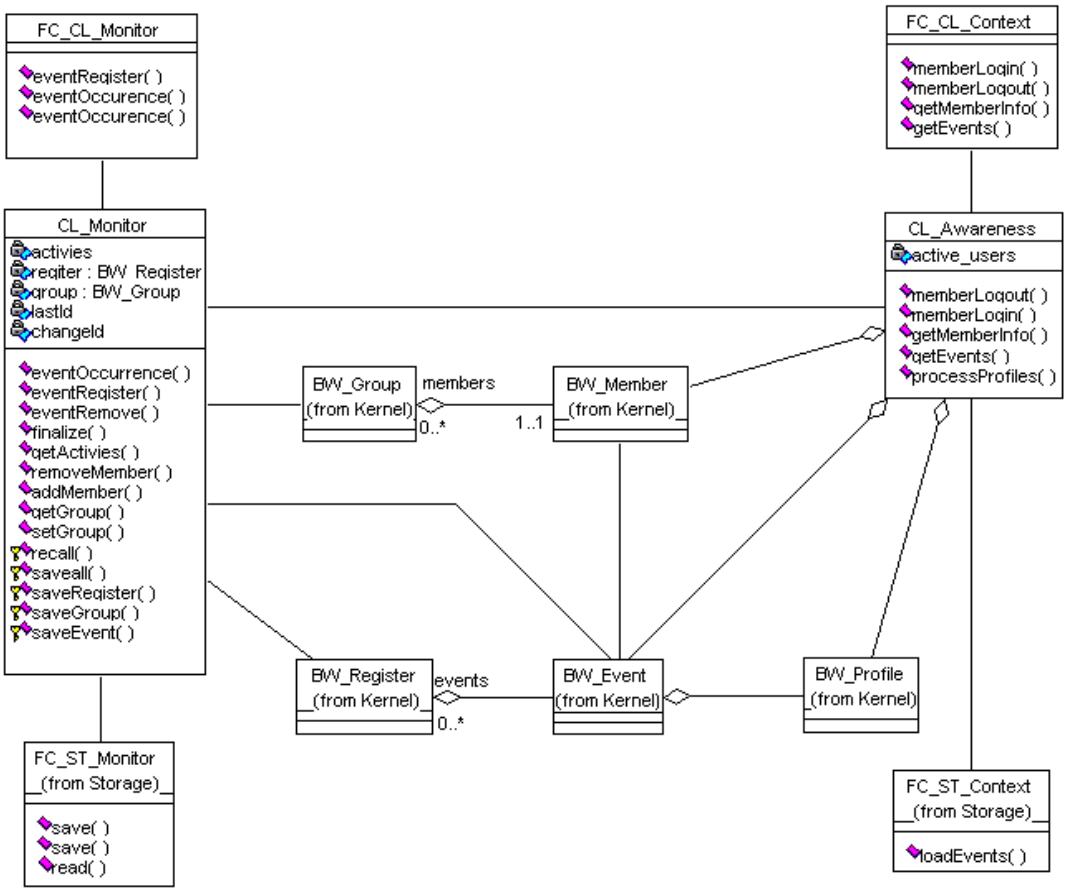


FIGURA 6.23 - Diagrama de classes do pacote *Control*

Desta forma, a classe “CL_Monitor” lida com todas as requisições oriundas do processo de monitoramento das atividades realizadas dentro no *groupware*. Este processo inclui o registro dos tipos de eventos que serão monitorados, realizado principalmente pelo método “eventRegister”, e o processo de ocorrência destes eventos, à medida que as suas respectivas atividades forem sendo concluídas dentro do *groupware*, através de chamadas ao método “eventOccurence”. Retornando aos diagramas de seqüência apresentados na Fig 6.3 e na Fig. 6.4, é possível se observar a presença de um objeto “CL_Monitor” realizando justamente estas atividades, o registro de um novo tipo de evento (Fig. 6.3) e a ocorrência de um evento específico (Fig. 6.4). Além destas funções, a classe “CL_Monitor” também realiza a manutenção do grupo, incluindo métodos para adicionar e remover membros do grupo (“addMember” e “removeMember”).

Já a classe “CL_Awareness” é responsável por todo o processamento referente à contextualização destes membros. É ela que controla a entrada e a saída de membros (métodos “memberLogin” e “memberLogout”) e a sua contextualização, após seu regresso ao sistema depois de um período de ausência ou após uma requisição explícita, pelo método “getEvents”. Esta contextualização, que representa a última etapa do Quadro 6.1, a notificação, dispara todo um processo interno à classe “CL_Awareness”. Este processo envolve a filtragem e o resgate dos eventos ocorridos e de interesse do usuário junto ao pacote *Storage,* e culmina com a apresentação ao usuário, via pacote *Interface*, como demonstrou o diagrama de seqüência da Fig. 6.6, apresentado no início deste capítulo.

Este processo de filtragem dos eventos ocorridos é um dos principais dentro do *framework* BW e envolve inicialmente a requisição dos *profiles* do usuário. Estes são obtidos junto ao objeto da classe “BW_Member” que o representa, através da chamada ao método “getProfiles”. Com o resultado desta chamada, a classe “CL_Awareness” chega ao conjunto de *profiles* ativos do usuário, incluindo o seu *profile* pessoal, o *profile* do papel que estiver sendo desempenhado no momento, e também o *profile* pessoal para este mesmo papel. Com estes três *profiles,* a classe “CL_Awareness” pode gerar o conjunto de tipos de eventos que são de interesse do usuário. Este conjunto é formado pela união de todos os tipos incluídos em cada um destes *profiles*, como mostra a Fig. 6.24 abaixo. Nesta figura são utilizados símbolos lógicos e matemáticos, tais como $\wedge$ (e lógico), $\rightarrow$ (implica), $\in$ (pertence) para a montagem do conjunto final, com todos os tipos de eventos de interesse do usuário, e do período total, com o período de tempo, seguindo a descrição de intervalos de Allen (ver Capítulo 5), cujos eventos ocorridos são de interesse desse usuário.

É possível se observar, então, na Fig. 6.24, que um tipo de evento estará presente no conjunto final “P_f” se estiver presente em, no mínimo, um dos três *profiles* ativos (“P_s”, “P_p” ou “P_{sp}”). Este conjunto final será utilizado para solicitar ao pacote *Storage* os eventos ocorridos dentro do período total “T_f” que forem de algum dos tipos do conjunto. O período “T_f” também é formado pela união dos intervalos de tempo relacionados a cada um dos *profiles,* de modo que o seu limite inferior será o menor limite inferior entre os intervalos de cada *profile,* e o intervalo superior, o maior deles, como mostra a Fig. 6.24. Com isso, o que é solicitado para o pacote *Storage* são todos os eventos ocorridos que são de interesse do usuário, ou seja, que satisfazem o seu conjunto de *profiles* ativos. Estes eventos são, então, passados para o pacote *Interface,* que os apresentará finalmente para o usuário.

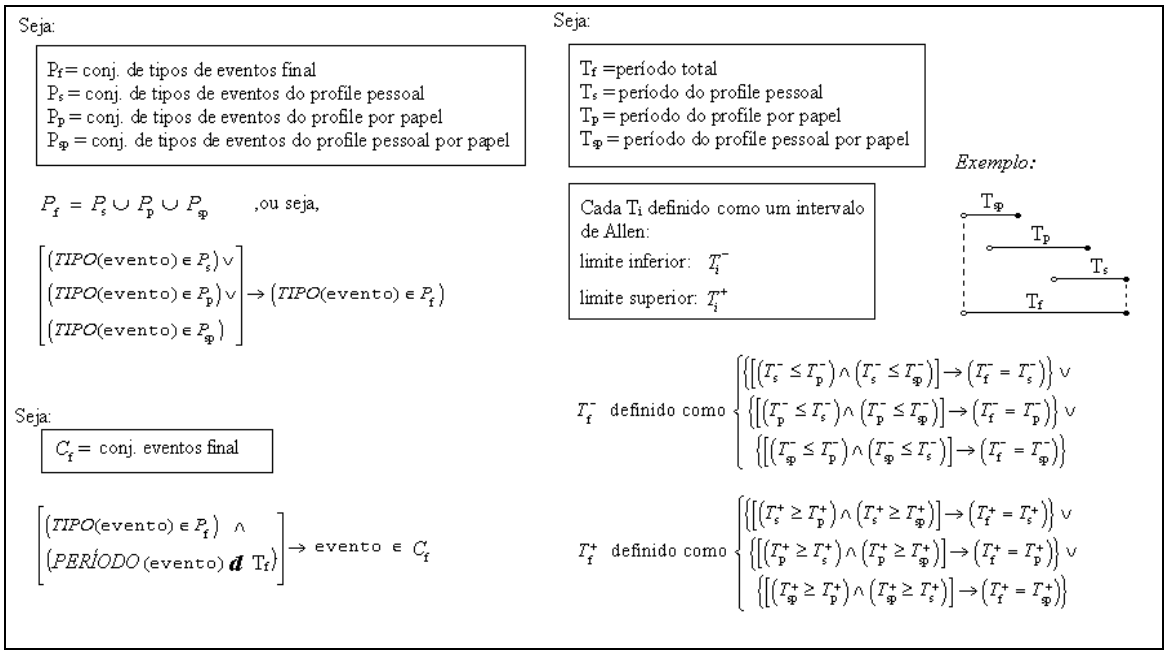


FIGURA 6.24 - representação matemática do processo de filtragem

Além das classes “CL_Monitor”, responsável pelos serviços de registro e ocorrência de eventos, e “CL_Awareness”, responsável pela filtragem, é possível se observar no diagrama de classes da Fig. 6.23 a presença de quatro classes de fachada: duas oriundas do pacote *Storage* (“FC_ST_Monitor” e “FC_ST_Context”) e duas do próprio pacote *Control* (“FC_CL_Monitor” e “FC_CL_Context”). Estas classes seguem a política de fachadas discutida na seção 6.2.2, onde cada pacote é definido contendo duas classes de fachada para sua comunicação com o pacote representante da camada superior, e ainda é capaz de visualizar duas outras classes de fachada, definidas no pacote da camada inferior. No caso do pacote *Control*, é exatamente esta a situação. As classes “FC_ST_Monitor” e “FC_ST_Context” pertencem ao pacote *Storage*, o qual representa a camada de armazenamento, colocada abaixo da camada de controle na arquitetura. Estas duas são as únicas classes do pacote *Storage* visíveis a *Control* e é somente através delas que se dá toda a comunicação entre os dois pacotes. Esta comunicação ocorre geralmente da seguinte forma: todas as requisições para salvar as informações resultantes do monitoramento (registro e eventos ocorridos) são feitas para a classe “FC_ST_Monitor”, enquanto todas as requisições para o resgate de eventos para a contextualização de um usuário são feitas para a classe “FC_ST_Context”.

Já as classes de fachada “FC_CL_Monitor” e “FC_CL_Context” têm funções semelhantes às classes “FC_ST_Monitor” e “FC_ST_Context”, mas são responsáveis pela comunicação entre o pacote *Control* e o pacote *Interface*, que realmente faz a notificação do usuário, e entre o próprio *framework* e o *groupware*. A classe “FC_CL_Monitor” recebe todas as requisições vindas do processo de monitoramento, como a ocorrência de um evento (método “eventOccurence”), normalmente originada do próprio *groupware*, enquanto a classe “FC_CL_Context” faz o mesmo pelo processo de contextualização, recebendo os pedidos normalmente originados do pacote *Interface*. Ambas, ao receberem estas requisições, repassam-nas aos seus respectivos responsáveis, à classe “CL_Monitor”, que tratará das requisições de monitoramento, e à classe “CL_Awareness”, para as requisições de contextualização.

Através destas quatro classes de fachada, todo o fluxo de informações é dividido em dois. O primeiro é voltado para o monitoramento, e normalmente tem sua origem nos serviços solicitados pelo *groupware* que são tratadas pelo pacote *Control* e, se necessário, transformados em requisições de armazenamento para o pacote *Storage*, fazendo com que as informações fluam de uma camada superior para as inferiores. E o segundo fluxo é voltado para a notificação, com a grande maioria das informações sendo resgatadas das camadas inferiores (pacote *Storage*) e chegando até às superiores (pacote *Interface*). Esta divisão do fluxo de informações objetiva evitar que houvesse um *overhead* no *framework* pela passagem de todas as informações trocadas entre dois pacotes por uma única classe. Com esta divisão em dois fluxos, toda a troca de informações é dividida sempre entre duas classes de fachada, organizando também este tráfego de acordo com seu objetivo, se de monitoramento ou de contextualização.

Estas duas fachadas são as únicas classes internas ao pacote *Control* visíveis tanto aos demais pacotes quanto ao próprio *groupware*. Esta pequena visibilidade externa vem ao encontro da idéia de independência entre os pacotes. Cada um dos três pacotes representantes das três camadas da arquitetura somente são capazes de visualizar as classes de fachada de seus pacotes vizinhos, o que torna cada um deles insensível a possíveis alterações feitas internamente aos vizinhos. A única exceção a esta regra é o pacote *Kernel*, que representa os dados manipulados por todos os demais pacotes. Estes são obrigados a ter conhecimento sobre o interior do pacote *Kernel*, pois devem ser capazes de lidar com as classes de informações ali definidas. Assim, pode-se observar dentro do próprio pacote *Control* esta visibilidade do pacote *Kernel*, através da presença de classes como “BW_Register”, “BW_Event”, “BW_Member” e “BW_Profile”.

6.2.4 Pacote Interface

O pacote *Interface* reflete dentro do *framework* a camada homônima da arquitetura. Seu objetivo, assim como o da camada de interface, é apresentar aos usuários as informações resultantes de todo o processamento anterior. É o pacote *Interface* o responsável pela notificação dos participantes, apresentando-lhes o resultado do processo de filtragem, que são os eventos ocorridos de seu interesse. Trata-se, portanto, da única ligação direta entre *framework* BW e os usuários propriamente ditos. Todas as demais são feitas através do *groupware*.

Para realizar o processo de notificação, o pacote *Interface* conta com um modelo de classes simples, apresentado na Fig. 6.25 abaixo. Percebe-se nesta figura a presença de algumas classes do pacote *Kernel*, responsáveis pela descrição das informações que serão apresentadas ao usuário através da interface, destacando-se os eventos, classe "BW_Event", o próprio membro, "BW_Member" e a superclasse "BW_Element", que descreve a base para todas estas informações. Também é possível perceber a presença da fachada para o pacote *Control*, a classe "FC_CL_Context", através da qual virão as informações processadas e resgatadas pelos demais pacotes. A presença destas classes descritas nos pacotes anteriores reforça as idéias da abertura do pacote *Kernel* e da visualização das classes de fachada. Somente o conteúdo do pacote *Kernel* é conhecido pelos outros pacotes, pois é ele que descreve as classes manipuladas por estes outros. Enquanto isso, entre si, os três pacotes representantes das três camadas da arquitetura só

vêem as classes de fachada dos demais, como o pacote *Interface* que só visualiza a classe de fachada "FC_CL_Context" de seu vizinho, o pacote *Control*, representante da camada de controle.

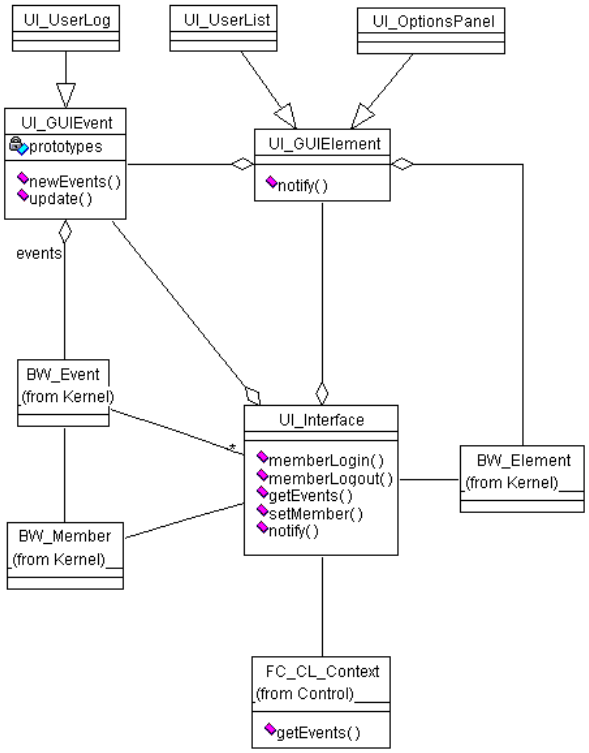


FIGURA 6.25 - Diagrama de classes do pacote Interface

Além destas classes oriundas de outros pacotes, a Fig. 6.25 destaca a presença da classe "UI_Interface". Esta classe centraliza todo o processamento por trás da correta apresentação dos dados. É ela quem faz o controle do ingresso de um usuário no sistema, solicita aos demais pacotes as informações sobre os eventos ocorridos, e direciona essas informações para as classes que realmente irão apresentá-las ao seu usuário final. Estas classes são "UI_GUI_Event" e "UI_GUI_Element" e ambas representam os elementos de interface que apresentarão as informações. A primeira, "UI_GUI_Event", é responsável por apresentar os eventos propriamente ditos, sendo que cada objeto pode lidar com um tipo de evento em específico. A segunda classe, "UI_GUI_Element", representa um elemento de interface completo, o qual irá agrupar os eventos apresentados pelas instâncias de "UI_GUI_Event". Assim, o controle sobre a apresentação de cada um dos eventos ocorridos individualmente é feito pela classe "UI_GUI_Event", enquanto que a apresentação do conjunto completo é realizada através da classe "UI_GUI_Element".

Cada uma destas classes é, na verdade, uma classe abstrata que apenas descreve as formas de comunicação entre o *framework* BW e a interface gráfica com o usuário. Com isso, estas duas classes devem ser especializadas para a construção da interface definitiva com o usuário. Esta interface deve ser adaptada ao *groupware* onde estiver inserida, fazendo com que o usuário perceba apenas a existência do *groupware* e não possa diferenciá-lo do *framework* BW. Para tanto, esta interface deverá ser criada obrigatoriamente pelo próprio desenvolvedor que estiver adaptando o *framework* BW para operar junto a sua ferramenta de *groupware*, pois somente ele será capaz de prever

as necessidades do sistema quanto à percepção e também as necessidades desta interface para o correto suporte a *awareness*. Isso, aliado as demais capacidades de expansão colocadas dentro do *framework* BW, como o caso dos eventos, permite a criação de elementos de interface específicos para o *groupware* tratado, através da especialização da classe "UI_GUI_Element", acompanhando as possíveis especializações da classe "BW_Event", e também permite agrupar estes tipos de eventos específicos da aplicação em uma representação gráfica adequada, através da criação de subclasses de "UI_GUI_Event". A Fig. 6.25 sugere, por exemplo, a criação de uma subclasse de "UI_GUI_Element", chamada "UI_UserList", para a apresentação da lista de usuários pertencentes ao grupo, e uma subclasse de "UI_GUI_Event", chamada "UI_UserLog", para, por exemplo, representar dentro desta lista os eventos resultantes do ingresso de usuários no sistema. Com estas subclasses e mais, provavelmente, uma subclasse de "BW_Event" específica para este tipo de evento, é possível, por exemplo, fornecer o histórico de entradas no sistema dos colegas, dando ao usuário percepção sobre como tem sido a presença de seus colegas no sistema.

Outro exemplo que pode ser considerado é o caso de um *groupware* de edição cooperativa que possua como tipo de evento específico (especialização de "BW_Event") a alteração sobre um fragmento de texto. Sempre que um dos participantes conclui sua modificação no fragmento, um evento deste tipo é gerado e enviado ao *framework* BW integrado a este *groupware*. Posteriormente, quando um outro usuário, que possua em um de seus *profiles* este evento, regresse ao sistema, será notificado quanto ao evento ocorrido. Imaginando, então, que a classe "UI_GUI_Element" foi especializada em uma outra classe que busque apresentar as informações de percepção sobre uma miniatura do texto, poder-se-ia ter uma subclasse de "UI_GUI_Event" que representasse este tipo específico de evento, mostrando as informações de cada uma das instâncias dos eventos ocorridos colorindo na miniatura os fragmentos alterados.

A divisão da representação gráfica permite que somente uma família de classes, a "UI_GUI_Event" e suas subclasses, mantenha as informações específicas sobre os eventos ocorridos, enquanto outra, "UI_GUI_Element", mantém a visão de um elemento gráfico completo e não as informações exatas por ele apresentadas. Juntas, estas classes podem funcionar como uma relação *container* e componente, como um painel completo e os itens de informação colocados neste. Desta forma, torna-se possível a criação de elementos de interface adequados e integrados ao *groupware*, através de subclasses de "UI_GUI_Element", que ainda assim contenham uma representação também adequada dos eventos ocorridos e seus tipos, através da especialização da classe "UI_GUI_Event".

Independente das subclasses criadas durante a adaptação do *framework* BW a um *groupware*, a comunicação destas com a classe "UI_Interface" e o restante do *framework* é feita da mesma forma, seguindo as interfaces definidas nas superclasses "UI_GUI_Element" e "UI_GUI_Event". Por exemplo, durante o processo de notificação, apresentado no início deste capítulo na Fig. 6.6, toda a comunicação é feita através de chamadas padrão, como "notify" e "update", fazendo com que a classe "UI_Interface" desconheça se os objetos com os quais se comunica são realmente das classes "UI_GUI_Element" e "UI_GUI_Event", ou quaisquer subclasses destas.

Todavia, é importante que estas subclasses, criadas para representar graficamente os eventos ocorridos, sejam corretamente projetadas. Estas são as principais responsáveis pela apresentação das informações oriundas do mecanismo de suporte a *awareness*, implementado pelo *framework* BW. Se estas classes não forem projetadas adequadamente, todo o funcionamento do *framework* estará comprometido, pois o

usuário poderá não perceber corretamente as informações que lhe forem apresentadas, incorrendo nos problemas discutidos na questão "como" no Capítulo 3. Logo, a fim de evitar que sejam criadas interfaces inadequadas, a próxima seção traz orientações para uma melhor construção destas, funcionando como uma espécie de pequeno guia para o desenvolvedor que estiver adaptando o *framework* BW ao seu *groupware.*

6.3 Orientações para a Construção da Interface

A interface com o usuário é um aspecto de grande importância para qualquer *software.* A interface é o meio através do qual o usuário se comunica com o sistema. É, para muitos usuários, o próprio sistema. Sua forma e seu conteúdo possuem uma forte influência em como este usuário verá e compreenderá as funcionalidades do sistema, podendo determinar o sucesso de uma interação sua com este. Sem interfaces especializadas, as taxas de aceitação destas aplicações permanecerão baixas, a sua utilização será difícil e os ganhos e benefícios permanecerão pequenos se comparados ao seu potencial [HIX 93],[PRE 95],[SOH 98].

Logo, da mesma forma que um *software* qualquer, uma ferramenta de *groupware* e o próprio mecanismo de suporte à percepção colocado neste, também dependem fortemente da interface utilizada. Esta será determinante no processo de assimilação da informação de *awareness*. Se esta interface não for adequada ao tipo de informação apresentada, o processo de notificação dos eventos ocorridos pode não surtir o efeito esperado, deixando o usuário sem perceber a informação desejada.

Exatamente devido a esta destacada importância da interface para o sucesso do mecanismo de suporte à percepção é que foi proposto este conjunto de sugestões de construção. Estas sugestões devem ser utilizadas por aquela pessoa que estiver introduzindo o *framework* BW em uma ferramenta de *groupware*, tipicamente o seu desenvolvedor, com a finalidade de produzir uma interface adequada ao tipo de grupo e usuário considerados “alvo” e compatível com o *groupware* utilizado.

Esta idéia de passar a construção da interface com o usuário para o desenvolvedor, e não incluí-la dentro do próprio *framework* BW, vem da busca por flexibilidade proposta neste trabalho. Isto pois a interface de um sistema, seja ele qual for, deve ser totalmente voltada ao seu usuário, para as tarefas que deverão ser executadas por ele. Citando Hix e Hartson [HIX 93]: “o usuário não deve ter que se adaptar à interface, mas ao invés disso, a interface deve ser projetada para ser intuitiva e natural para o usuário aprender e usar”. Ou seja, é necessário se conhecer o usuário e o grupo para melhor projetar a interface do *groupware* e do mecanismo de suporte à percepção para este.

Portanto, não seria possível incluir dentro do *framework* BW uma interface predeterminada sem antes fixar, no mínimo, quem seria o usuário, como seria o seu grupo de trabalho e quais seriam as suas atividades. Isto inviabilizaria o objetivo de manter este trabalho flexível a ponto de ser incluído em mais de uma ferramenta de *groupware*. Sendo assim, optou-se por definir apenas uma interface abstrata, vista na seção anterior, a qual deve ser obrigatoriamente especializada pelo desenvolvedor. Este desenvolvedor detém o conhecimento sobre quem é este usuário, quem é o grupo e que atividades serão realizadas. Somente ele poderá criar uma interface voltada para este usuário, tanto no tocante ao *groupware* quanto ao mecanismo de suporte à percepção em si.

Todavia, para facilitar o trabalho de criação da interface com o usuário para o *framework* BW, esta seção propôs um conjunto de orientações ou sugestões que deverão ser seguidas pelo desenvolvedor, durante o processo de especialização das classes do pacote "Interface". Este conjunto de orientações foi elaborado com base em alguns estudos dentro da área de Interação Homem-Máquina, mais conhecida por HCI _ *Human-Computer Interaction*, especialmente *guidelines* e trabalhos sobre *usability*. A área de HCI busca entender como as pessoas interagem com sistemas de computador, de modo que melhores sistemas, os quais atendam às necessidades dos usuários, possam ser projetados. Este é o objetivo de HCI: desenvolver e melhorar sistemas para que os usuários possam realizar suas tarefas com segurança (*safety*), eficiência (*efficiently*), de modo efetivo (*effectively*) e agradável (*enjoably*). Estes quatro aspectos juntos são conhecidos como usabilidade (*usability*). Usabilidade pode ser considerada como a combinação de diversas características voltadas para o usuário: fácil aprendizado, alta performance para o usuário realizar sua tarefa, com baixa taxa de erros, satisfação e retenção do usuário sobre o tempo [HIX 93],[PRE 95].

Esta interação estudada dentro de HCI consiste em quatro componentes básicos: o próprio usuário, a tarefa que ele deve executar, o contexto particular onde ele está inserido e o sistema que será usado por ele [PRE 95]. Aqui, foi incluída uma complexidade a mais nesta interação, o grupo. Quando em um ambiente cooperativo, o usuário não mais interage sozinho com o sistema, ele também deve interagir, através do sistema, com os demais colegas de grupo. Desta forma, o projeto da interface dentro de uma ferramenta de *groupware* deve sempre considerar que estão interagindo com o sistema vários usuários, os quais também devem interagir entre si, síncrona ou assincronamente, e que juntos realizam um conjunto de tarefas. Isto amplia completamente o contexto envolvido e torna este projeto bem mais complexo do que o já problemático projeto de interfaces mono-usuárias. Através das sugestões que serão apresentadas nesta seção, espera-se reduzir um pouco esta complexidade, auxiliando o desenvolvedor na tarefa de projetar esta interação entre o usuário e o mecanismo de suporte à percepção.

Este conjunto de orientações foi desenvolvido principalmente sobre *guidelines* para interfaces mono-usuárias. O termo *guideline* engloba tanto princípios amplos, que oferecem advertências gerais, quanto regras de projeto [PRE 95]. Todavia, este trabalho não tem a pretensão de ser um *guideline*, uma vez que a construção de *guideline*, mesmo que voltado somente de interfaces para o fornecimento de *awareness*, seria por si só uma outra dissertação. O que esta seção propõe é um simples conjunto de sugestões, baseadas nestes *guidelines*, adaptando-se este conhecimento sobre interfaces mono-usuárias para ambientes cooperativos, e no senso comum sobre o que seria necessário para uma apresentação adequada da informação de *awareness*. É claro que, se o desenvolvedor tiver algum conhecimento sobre o projeto de interfaces em geral ou sobre métodos de avaliação de usabilidade, este deve se sentir encorajado a utilizar estes conhecimentos para a especialização do pacote *Interface*.

Recomenda-se, caso o desenvolvedor esteja familiarizado, o uso de conceitos colocados em algumas abordagens modernas para o desenvolvimento de sistemas, tais como métodos participativos e *user-centred design* (projeto centrado no usuário), ou ainda a realização de testes de usabilidade. Os métodos participativos levam em conta requerimentos sociais e organizacionais nos primeiros estágios do ciclo de desenvolvimento. Os usuários participam analisando estes requerimentos sociais e planejando as estruturas sociais e técnicas para suportar tanto necessidades individuais como da organização. Os projetos centrados no usuário (*user-centred design*), por sua

vez, buscam tornar os usuários o foco das atividades de projeto, envolvendo-os e levando em conta suas necessidades por todo o projeto, enquanto os testes de usabilidade procuram fornecer informações para a atualização e manutenção de sistemas já existentes. Estes procuram analisar características como o tempo e o esforço para o aprendizado, o *throughput* do usuário, os erros cometidos e a flexibilidade de adaptação do sistema a novas interações de usuários mais experientes [PRE 95].

Os itens a seguir trazem, então, estas orientações para a construção da interface com o usuário para o *framework* BW, algumas das quais foram também utilizadas para o desenvolvimento do protótipo, discutido no próximo capítulo. Lembrando sempre que estas orientações apresentadas abaixo são baseadas em *guidelines* para interfaces mono-usuárias e no senso comum sobre ferramentas de *groupware*, mas não representam, de forma alguma, um *guideline*, ou um guia, completo e definitivo, para a construção deste tipo de interface. Reforçando, é apenas um despretensioso conjunto de sugestões para ajudar o desenvolvedor, o qual não deve também se sentir obrigado a utilizá-las todas, mas somente àquelas que melhor se encaixarem em suas necessidades.

♦ *Conhecer o Usuário e o Grupo:*

Os usuários não são um grupo homogêneo. Eles diferem uns dos outros fisicamente, em experiência e conhecimento, psicologicamente e sócio-culturalmente. Estes fatores podem influenciar em como um indivíduo irá lidar com o sistema. Além disso, grande parte do trabalho envolve grupos de pessoas interagindo entre si e com vários objetos. Cada organização tem sua cultura própria que pode, por sua vez, ser afetada por sistemas de computador. Assim, os aspectos sociais de CSCW transformam-se em um importante tópico de pesquisa. Para que ferramentas de *groupware* tenham sucesso, é necessário compreender diversos aspectos, tais como o ambiente organizacional, a forma de comunicação entre as pessoas, o que estas pessoas querem que a tecnologia faça por elas [PRE 95]. Em outras palavras, é muito importante conhecer e caracterizar o usuário que irá trabalhar com o sistema, saber em que contexto ele está inserido, quem é seu grupo de trabalho, qual sua estrutura e quais serão as suas atividades.

Para Hix e Hartson [HIX 93], conhecer o usuário significa compreender o comportamento humano. Significa conhecer as características dos usuários que irão utilizar uma interface em particular, pois o projeto desta interface deve ser adaptado para facilitar o uso que estes usuários irão fazer do sistema. E conhecer o usuário inclui também ser simpático as suas diferentes necessidades, fornecendo, por exemplo, atalhos de programas para usuários mais experientes e permitindo-lhes realizarem suas tarefas de mais de uma maneira [PRE 95]. Desta forma, é bastante interessante acrescentar, mesmo na interface de uma ferramenta de *groupware* recursos para facilitar o trabalho dos usuários, como atalhos, e para tornar a ferramenta mais flexível e adaptável, sempre que possível.

♦ *Respeitar o Elemento Humano:*

Quando as pessoas interagem com um sistema de computador, elas estão primariamente interagindo com informação, e a interação homem-máquina envolve o processamento desta informação na mente. É necessário garantir que este processamento esteja dentro das capacidades de processo mental do usuário. Logo, ao se projetar uma tela é importante garantir que a informação esteja legível, distinguível, compreensível, ordenada e significativamente estruturada [PRE 95]. Em outras palavras, o projeto de uma interface deve ser todo voltado para o elemento humano da interação, e o sucesso desta interface junto a este elemento depende do processo de assimilação da informação que lhe é apresentada. Esse processo de assimilação envolve o processamento mental desta informação pelo usuário, o qual pode ser facilitado à medida que esta informação é apresentada de forma estruturada, ordenada , legível e facilmente distinguível do fundo ou das demais informações. Portanto, estas características devem ser buscadas com afinco pelo desenvolvedor que estiver criando a interface para o *framework* BW, uma vez que é importante o usuário assimilar corretamente as informações de *awareness.*

♦ *Utilizar a Memória Sem Causar Sobrecarga:*

A memória está envolvida em todas as ações humanas, mas a habilidade de lembrar de coisas é altamente variável. Algumas operações simples exigem um esforço mínimo para memorizá-las, enquanto outras exigem contínuo aprendizado e seguidamente saem da memória logo após terem sido usadas. Isto conduz a necessidade de se reduzir a carga cognitiva, projetando a interface de modo que os usuários não precisem lembrar de um número muito grande de detalhes. Isto inclui minimizar a memorização e o aprendizado e, principalmente, minimizar a quantia total de informações, apresentando somente aquilo que é necessário para o usuário. Tornar uma representação visual mais complexa do que necessário, apenas porque é possível, pode na verdade mais atrapalhar um usuário a completar a sua tarefa. [PRE 95],[HIX 93].

Portanto, é importante manter a interface simples. Tarefas simples podem ser mantidas mais facilmente usando ações, ícones e outros objetos que forem naturais para o usuário. A idéia é limitar o número de itens com os quais um usuário precisará lidar em um momento em particular. Uma boa maneira de fazer isso é permitir que o usuário reconheça as informações, ao invés de ter que relembrar, sempre que possível. Esta, por sinal, é uma das descobertas mais estáveis na pesquisa sobre memória: o ser humano pode reconhecer um material em uma tela bem mais facilmente do que se tivesse que relembrá-lo não olhando para a tela [HIX 93],[PRE 95].

Este problema da sobrecarga cognitiva já foi abordado neste trabalho dentro do Capítulo 3, ao se discutir as questões "como" e "quanto". Conforme se discutiu naquela ocasião, há, dentro do estudo sobre *awareness,* uma preocupação constante quanto à quantidade e qualidade de informação apresentada ao usuário. Se apresentadas poucas informações de *awareness,* este usuário pode ficar sem o contexto necessário para acompanhar as atividades realizadas dentro do grupo. Se apresentadas muitas

informações, o usuário poderá se sentir soterrado, despender atenção em demasia para analisar estas informações, e corre o risco de não perceber aqueles eventos que lhe forem mais importantes. E ainda, se estas informações forem apresentadas de forma inadequada, com uma interface ineficiente, de difícil aprendizado e que exija uma forte memorização do usuário, este não se sentirá atraído a utilizar o mecanismo de suporte à percepção e poderá não ser capaz de assimilar as informações de *awareness* de que precisa.

Desta forma, é extremamente necessário minimizar a quantidade de informação apresentada a este usuário, para não sobrecarregá-lo. Este processo de minimização tem seu início dentro do próprio *framework* BW, no processo de filtragem através dos *profiles* do usuário. Este processo limita a apresentação dos eventos ocorridos a somente aqueles que forem de tipos de interesse do usuário, fazendo com que não sejam apresentados absolutamente todos os eventos ocorridos, mas apenas um subconjunto destes. Todavia, este processo de filtragem sozinho não é o suficiente. É muito importante que a apresentação destas informações de *awareness* não sobrecarregue por si só o usuário. Esta apresentação deve ser feita de forma simples e adequada ao tipo de evento relacionado. Isto significa adaptar a interface do mecanismo de suporte à percepção ao tipo de grupo e usuário que o estará utilizando, e também ao tipo de informação, os eventos, que serão apresentados, adotando elementos de interface que melhor os represente.

Para tanto, deve-se utilizar as técnicas, como as citadas em Preece [PRE 95] e em Hix e Hartson [HIX 93], para a minimização tanto da quantia total de informação, quanto da memória e do aprendizado. Esta técnicas incluem ser consistente, usar redação simples, instruções concisas e formato de dados familiares, utilizar seleção por menus ao invés de comandos, nomes ao invés de números, fornecer documentação, escolher nomes e símbolos significativos, manter ícones, mensagens e outros textos fáceis de se reconhecer e aproximar-se de outros sistemas já conhecidos. Também se deve, de preferência, utilizar elementos gráficos, incluindo ícones e menus, e procurar sempre manter o mesmo padrão de interface , envolvendo cores, formas, janelas, etc, utilizado dentro do próprio *groupware*, uma vez que este padrão provavelmente já será de conhecimento do usuário.

Além disso, deve-se evitar apresentar todos os detalhes sobre os eventos ocorridos de uma única vez. Deve-se dar preferência inicialmente a representações resumidas dos eventos, mantendo seus detalhes ocultos até que o usuário ativamente os solicite.

- *Manter a Consistência da Interface:*

Sempre se espera que coisas similares sejam feitas de formas similares. Na interface, para uma semântica similar, uma sintaxe similar deve ser usada, pois usuários vão esperar que certos aspectos da interface se comportem de certas maneiras e quando isto não ocorre, podem ficar confusos. Logo, é necessário manter consistência na interface [HIX 93]. Esta consistência emerge de operações e representações padrão, e do uso de metáforas apropriadas. O uso de padrões busca chegar à consistência através de produtos que são do mesmo tipo. Eles encorajam a formação de uma terminologia comum, de uma identidade comum, tornando sistemas com um mesmo *"look and feel"*

facilmente reconhecidos, e resultam em uma redução no treinamento [PRE 95]. Trazendo diretamente para a realidade deste trabalho, um desenvolvedor, ao incluir o *framework* BW em seu *groupware*, deverá tomar cuidado para projetar a interface deste mecanismo de *awareness* consistente com a utilizada dentro do próprio *groupware*. Deve-se procurar sempre integrar estas duas interfaces como se fossem uma só, um único produto, utilizando-se para isso os mesmos padrões usados dentro do próprio *groupware*.

Manter consistência envolve a organização dos dados na tela. Para tanto, é importante fazer o seu *layout* de modo a permitir aos usuários saber onde encontrar uma dada informação. Simplesmente organizando a informação de forma acessível pode reduzir drasticamente o tempo levado para encontrar uma informação específica. Agrupar itens similares na tela aumenta também a legibilidade e pode destacar relacionamentos entre tipos diferentes de informação. Para isso, é possível utilizar técnicas como a codificação por cor, o uso de bordas gráficas entre os diferentes grupos de informação, destaque através de vídeo reverso e brilho [PRE 95].

Em outras palavras, deve-se procurar manter sempre em uma mesma localização na interface os elementos que notificam o usuário sobre as informações de *awareness*, para que este possa facilmente chamar estas informações sempre que desejar. Também se deve procurar agrupar estas informações de acordo com os tipos de eventos correspondentes. Para tanto, aconselha-se a criação de subclasses de "UI_GUI_Element" de modo a reunir os eventos relacionados, agrupando-se nestas as subclasses de "UI_GUI_Event" correspondentes a estes tipos de eventos relacionados. Assim, pode-se organizar melhor as informações, agrupando os eventos similares em um mesmo elemento de interface, facilitando, como conseqüência a assimilação da informação de percepção.

Um exemplo de uso desta sugestão pode ser visto no protótipo discutido no próximo capítulo, no qual foi incluído, dentro da própria estrutura de menus do *groupware*, uma chamada direta à interface do *framework* BW. Esta chamada foi incluída da mesma forma e no mesmo menu que outros recursos de percepção disponíveis no *groupware*, mantendo, assim, a consistência da interface.

- *Utilizar Metáforas*

Para tornar o aprendizado do sistema mais fácil, deve-se explorar métodos de ensino ativos, que auxiliam neste processo. Estes métodos incluem o uso de manuais reduzidos e orientados a tarefas, além do uso de metáforas. Estas últimas representam um modelo concreto com o qual o usuário já está familiarizado. O problema, entretanto, é decidir qual metáfora será a mais adequada a uma tarefa. O fator chave para esta decisão é procurar garantir que ações, procedimentos e conceitos deste domínio familiar ao usuário sejam mapeados o mais próximo possível das estruturas da aplicação representadas na interface. Assim, o uso de situações, vocabulário, desenhos e metáforas naturais e conhecidas pelo usuário ajuda a tornar previsíveis as interações do usuário com o sistema, diminuem o processamento mental que este usuário precisará fazer sobre as informações, reduzindo a carga cognitiva sobre ele [PRE 95], [HIX 93].

Neste sentido, pode-se conseguir reduzir a carga cognitiva sobre o usuário através do uso, dentro do mecanismo de suporte à percepção, de uma metáfora semelhante ou que melhor se integre à utilizada dentro do *groupware*. A idéia é fazer com que o usuário naturalmente se adapte com a utilização de um novo mecanismo de *awareness*, à medida que este mecanismo não é percebido como um outro sistema além do *groupware*. É muito importante que o usuário não sinta diferença entre as informações fornecidas somente pelo *groupware* e aquelas manipuladas e apresentadas pelo *framework* BW. Fazer com que este usuário veja somente um sistema reduzirá o tempo de aprendizagem, a taxa de erros e, principalmente, a carga cognitiva sobre este usuário, já que ele poderá ignorar o processamento feito para trazer as informações de *awareness* até ele.

- *Chamar com Cautela a Atenção do Usuário*

Simplesmente disponibilizar as informações de *awareness* para o usuário não garante que ele as tenha percebido. É importante chamar a atenção deste usuário para as informações vindas do mecanismo de suporte à percepção. Todavia, deve-se ter muita cautela ao chamar sua atenção, pois o uso excessivo de técnicas para isto pode facilmente mais irritar o usuário do que ajudá-lo.

Direcionar a atenção do usuário para uma parte específica de informação pode ser obtido de diversas formas. Algumas destas formas incluem o uso de marcadores visuais como texto em negrito ou sublinhado, piscante, vídeo reverso, cores e até mesmo avisos sonoros. Além disso, pode-se usar outras técnicas como apresentar a informação em uma estrutura lógica e significativa, dividir a tela em seções ou janelas discretas e sobrepostas, que podem ser facilmente associadas a informações específicas, permitindo que os usuários executem múltiplas tarefas [HIX 93], [PRE 95]. Dessa forma, deve-se aproveitar a possibilidade de agrupamento lógico da informação de *awareness* que o *framework* BW oferece através da especialização do pacote interface, relacionando os tipos de eventos com os elementos de interface que os apresentam, por exemplo, através de janelas distintas, com destaques também diferenciados, de acordo com a importância que estes eventos podem ter para o trabalho do grupo como um todo. Estes elementos podem ainda utilizar recursos como cores, ícones, gráficos, áudio e menus, comentados a seguir, para melhor direcionar a atenção do usuário.

- *Usar Adequadamente Cores*

As cores podem ser muito efetivas para segmentar uma tela ou janela em regiões separadas, podem facilitar tarefas de pesquisa e detecção de informações, e ainda melhorar a legibilidade de um símbolo colorido sobre o fundo. As cores podem ser usadas para mostrar relacionamentos entre objetos, para efetivamente chamar a atenção do usuário para uma informação importante ou que se altera com freqüência e também para tornar objetos, como ícones, mais realistas. Todavia, as cores devem ser usadas com cuidado e significado. Muitas cores embaralham a tela, aumentando o tempo de pesquisa por uma informação. Por exemplo, não se deve utilizar mais de quatro cores diferentes em uma única tela de texto Além disso, certas combinações de cores, como

vermelho sobre azul, devem ser evitadas. Outro exemplo é o uso de azul para texto. Este é inadequado para textos, já que é uma das cores mais difíceis de ser lida, pois os receptores de cores do olho humano são particularmente insensíveis ao azul, fazendo dele uma ótima cor para fundo, mas não para o texto. Outro ponto a se considerar é a presença de usuários com dificuldade visual, como o daltonismo, para os quais o uso de cores pode ser perdido [HIX 93],[PRE 95].

- *Usar Adequadamente Ícones*

Ícones são pequenas imagens gráficas comumente usadas para representar diferentes aspectos de uma metáfora de interface. Estes aspectos incluem objetos, opções, aplicações e mensagens. A vantagem do uso de ícones é que, em muitos casos, eles são mais fáceis de aprender e lembrar, pois fornecem mais informações visuais sobre o objeto, agindo como uma forte ligação mnemônica e explicitamente mostrando os relacionamentos entre objetos do sistema [PRE 95].

Todavia, ao se projetar um ícone, é importante considerar o contexto no qual ele será usado, pois este irá influenciar na compreensão do ícone. Também é importante considerar o domínio da tarefa para o qual o ícone será usado (algumas tarefas são mais suscetíveis à representação gráfica), a forma usada para representar o objeto, a natureza deste objeto representado e até onde o ícone poderá ser separado dos demais. É importante considerar que uma interface pode se tornar rapidamente confusa e ocupada pelo uso de muitos ícones. Nestes casos, quando só o desenho não é o bastante, um *label* deve ser adicionado ao ícone, para que o usuário possa distingui-lo com maior facilidade [HIX 93],[PRE 95].

- *Usar Adequadamente Textos*

Existem vários *guidelines* voltados para o uso de texto. Muitos destes incluem considerações como as encontradas em Preece [PRE 95] e Hix e Hartson [HIX 93]: texto convencional, em caixa alta (*uppercase*) e caixa baixa (*lowercase*) pode ser lido em torno de 13% mais rápido que o texto todo em caixa alta; caracteres em caixa alta são mais efetivos para itens que precisam atrair a atenção; texto "piscante" deve ser usado para atrair a atenção apenas para eventos de missão crítica; não se deve usar mais de três fontes distintas em uma única tela etc.

Além disso, o próprio conteúdo do texto também deve ser direcionado para o usuário. Por exemplo, usar um vocabulário voltado para o usuário e à tarefa, ao invés de voltado somente ao sistema, contribuirá muito para sua comunicação [HIX 93]. Isto pode ser feito desde cedo junto ao *framework* BW, através do uso de um vocabulário compatível e compreensível pelo usuário na definição do texto nos campos de descrição e detalhes de cada evento ocorrido.

♦ *Usar Adequadamente Gráficos*

A representação por gráficos desempenha importante papel na apresentação da informação. Aplicações e tarefas mais suscetíveis a esta forma de representação incluem aquelas que requerem o julgamento visual do usuário e aquelas envolvendo dados complexos [PRE 95].

O fornecimento de *awareness* pode ser considerado uma destas aplicações que requerem o julgamento do usuário e que também envolve dados complexos. Desta forma, é aconselhável, sempre que possível, que sejam utilizados gráficos para apresentar a informação de percepção. Através de gráficos, é possível apresentar esta informação de forma mais organizada e resumida, permitindo a apresentação de uma visão geral dos eventos ocorridos durante o último período de ausência do usuário. Um exemplo da utilização desta sugestão é apresentada na Fig. 7.11 do próximo capítulo, onde um gráfico de barras colorido é usado para fornecer um rápido resumo de todos os eventos ocorridos.

Todavia, o projetista de um sistema deve decidir qual a melhor maneira de apresentar os dados, a maneira mais apropriada à tarefa em questão. Algumas das opções são as seguintes [PRE 95]:

1) Gráfico de linhas e curvas mostram como duas variáveis contínuas estão relacionadas, recomendando-se no máximo quatro linhas (curvas) por gráfico e, quando houver múltiplas linhas, utilizar um label adjacente para cada uma;
2) Gráficos de área, banda ou superfície podem ser usados quando muitas linhas ou curvas representam porções de um todo;
3) Gráficos de barras ou histogramas mostram os valores de uma única variável contínua para várias entidades separadas ou para uma amostra variada de intervalos discretos;
4) Gráficos de torta mostram a distribuição relativa dos dados entre partes que formam um todo, recomendando-se usar até cinco segmentos;
5) Medidores ou simuladores mostram o valor de uma variável contínua.

♦ *Uso de Recursos de Áudio*

O áudio pode ser usado como uma ligação para eventos importantes e é seguidamente efetivo como canal de saída redundante. Os toques suaves devem ser usados para fornecer um *feedback* positivo, e toques mais severos, para emergências ou para conseguir atenção imediata do usuário [ERF 99]. Todavia, não se deve abusar na utilização de sons junto à interface, pois o seu excesso ou o uso contínuo de tons desagradáveis ao ouvido humano, como tons muito agudos, podem causar desconforto ou irritação junto ao usuário.

♦ *Usar Adequadamente Janelas e Menus*

Janelas diferentes podem ser usadas para separar tarefas diferentes e para apresentar diferentes visões coordenadas de uma mesma tarefa. Entretanto, não se deve superutilizá-las. Também se deve manter a consistência em aparência e comportamento, especialmente na janela principal. Esta é uma âncora para o usuário, em especial, quando muitas janelas podem ser abertas, pois este usuário poderá se sentir perdido e querer retornar à janela principal como ponto de partida. Sempre que ele retornar a esta janela, ela deve estar como ele a deixou, exceto se alguma ação deste usuário em uma janela secundária tenha definido alterações na principal, ou se esta última apresenta um estado dinâmico. Além disso, cada nova janela criada deve aparecer em uma posição fixa e então permitir ao usuário reposicioná-la e redimensioná-la [HIX 93].

Preferencialmente, deve-se utilizar janelas, dentro do contexto deste trabalho, para dividir e organizar melhor as informações de *awareness.* Pode-se, por exemplo, resumir, por meio de gráficos ou ícones, em uma janela principal, o conjunto completo dos eventos ocorridos de interesse do usuário, fornecendo assim uma visão geral, e em janelas secundárias, solicitadas pelo usuário, via botões, ícones etc, fornecer maiores detalhes sobre os eventos agrupados por tipo, ou sobre um evento específico.

Além disso, pode-se utilizar menus para facilitar a interação com o usuário , deixando mais acessíveis o conjunto de ações possíveis ao usuário. Estes menus são, normalmente, simples descrições textuais de funções disponíveis, mas também podem consistir de ícones e botões, mostrados verticalmente ou horizontalmente, ou ainda consistir de caixas que podem ser marcadas pelo usuário. Todavia, tanto o nome das opções, quanto o conteúdo de ícones e a descrição dos botões devem ser auto-explicativos [PRE 95].

♦ *Cuidar as Mensagens de Erro*

Pessoas sempre cometem erros e de fato devem cometê-los para aprender. Todavia, deve-se tomar ações que previnam (ou, no mínimo, dificultem) o usuário de cometer um erro, fornecendo boas mensagens de erro e um grande número de diagnósticos explicativos. Estas mensagens devem ter um vocabulário positivo, não agressivo e voltado ao usuário, com termos específicos e construtivos. Estas mensagens também devem ser breves e concisas, por exemplo, apresentando uma breve descrição do erro e fornecendo a opção de ajuda para o usuário obter maiores informações [HIX 93],[PRE 95].

A prevenção de erros do usuário deve ser uma preocupação em qualquer tipo de *software* e dispensa maiores comentários. Todavia, esta abordagem de textos breves e concisos, com vocabulário não agressivo voltado ao usuário e opção para busca por maiores detalhes, deve ser adotada ao se apresentar de forma textual o conteúdo dos eventos ocorridos. Estes devem obrigatoriamente ter uma primeira apresentação resumida, não repleta com todos os detalhes. Estes detalhes, no entanto, devem estar disponíveis para aqueles usuários que desejem receber todos os detalhes, através de menus, ícones, janelas secundárias e outras opções.

Além disso, muitas interfaces gráficas ajudam o usuário a evitar erros desabilitando opções errôneas, por exemplo, acinzentando opções de menus e botões. Entretanto, esta abordagem pode gerar frustração em usuários que não sabem porque a opção desejada está indisponível. Para resolver isto, pode-se mostrar mensagens de ajuda, caso o usuário tente escolher uma opção assim. Também é apropriado fornecer ao usuário indicadores de *status* apropriados, especialmente quando o sistema estiver executando um processamento potencialmente lento [HIX 93].

- *Facilitar o Uso*

Ao se projetar uma interface, deve-se procurar facilitar seu uso, buscar o menor esforço para o usuário, oferecer, especialmente para aqueles usuários mais freqüentes, uma eficiência maior, através, por exemplo, de teclas de atalho e de funções, ou fornecendo diferentes visões de um mesmo objeto, Além disso, deve-se ajudar o usuário a começar no sistema. Fornecer um *overview* com as informações básicas, como convenções, funções básicas, como selecionar opções e onde procurar ajuda, é uma prática que ajuda o usuário a se tornar confortável com o sistema rapidamente [HIX 93].

No caso específico deste trabalho, é importante que o usuário se familiarize rapidamente com os novos elementos de interface referentes ao mecanismo de suporte à percepção. Para tanto, fornecer um *overview* sobre o novo mecanismo de *awareness,* informando para que servirá e como utilizá-lo, é vital, principalmente no tocante aos *profiles.* Deve-se obrigatoriamente oferecer um *overview* ou um mecanismo de ajuda que explique o que são os *profiles,* sua importância e como manipulá-los. É muito importante que o usuário esteja consciente que será através destes *profiles* que ele poderá expressar sua opinião sobre quais atividades realizadas dentro do *groupware* são de seu interesse.

- *Utilizar Widgets Específicos*

Para facilitar o trabalho do desenvolvedor durante a construção da interface para o *framework* BW, aconselha-se fortemente o uso ou a adaptação de *awareness widgets* sempre que estes estiverem disponíveis. "Um *awareness widget* é um objeto de interface com o usuário, como janelas e *scroolbars,* que é especificamente projetado para fornecer informação de *awareness"* (Prakash et al. [PRA 99]). Todavia, isto não exclui a necessidade de se analisar se algum dos *widgets* disponíveis realmente se encaixa na situação e no tipo de informação que se deseja apresentar.

- *Integrar Awareness e Groupware*

Já foi comentado anteriormente que a interface deve ser consistente. Isto envolve desenvolver a interface para o *framework* BW, utilizando os mesmos padrões, o mesmo "*look and feel"* da interface do *groupware,* de modo que o usuário perceba tudo como um único produto. Além disso, esta completa inserção da interface do *framework* BW na própria interface do *groupware* deve ser buscada pelo desenvolvedor por vários

outros motivos. O primeiro deles, é que a informação de *awareness* deve estar sempre disponível ao usuário, de forma simples e intuitiva, de modo que ele não precise buscá-la exaustivamente, mas a tenha sempre à mão (ou ao alcance de um *click*). Outro ponto importante é que a informação de *awareness* deve ser facilmente interpretada, deve ser apresentada em um contexto familiar, idealmente o próprio *workspace* [SOU 96].

Em resumo, a informação de *awareness* deve estar incorporada ao ambiente de trabalho dos usuários e ser facilmente acessível, e ainda ser automaticamente capturada e distribuída sempre que possível [PRA 99], e todas estas características só podem ser alcançadas dentro do escopo deste trabalho através da integração completa deste mecanismo de suporte à percepção à ferramenta de *groupware*. Um exemplo desta integração é comentado no próximo capítulo, onde o *framework* BW foi inserido no *groupware* COPSE/CUTE usando a mesma idéia de janelas independentes e no mesmo menu usado pelos demais recursos de percepção.

6.4 Considerações Finais

Este capítulo apresentou detalhes da estrutura e do funcionamento do *framework* BW. Este constitui a essência deste trabalho, formando um mecanismo de suporte à percepção de eventos no passado, flexível e adaptável, o qual poderá ser incluído em mais de uma ferramenta de *groupware*. Além disso, este capítulo também apresentou um conjunto de orientações para a construção da interface do *framework* com o usuário. Estas orientações objetivam auxiliar o desenvolvedor a criar uma interface adaptada ao seu grupo de usuários e à ferramenta de *groupware*, e que seja capaz de adequadamente apresentar as informações da *awareness*. Entretanto, este conjunto de orientações objetiva somente auxiliar o desenvolvedor, sem ter nenhuma pretensão de ser um *guideline* completo para isto.

Com isso, concluiu-se aqui toda a descrição e os aspectos teóricos deste trabalho, abordado sob diversos ângulos em detalhes. No entanto, ainda resta sua comprovação prática. Esta foi realizada através da construção de um protótipo simples, adaptando o *framework* BW a uma ferramenta de *groupware* já existente. A descrição completa de todo deste processo de prototipagem é feita no próximo capítulo.

7 Construção do Protótipo

Os capítulos anteriores apresentaram o mecanismo de suporte à percepção de eventos no passado proposto dentro deste trabalho. Este mecanismo é formado pelo *framework* BW, discutido no Capítulo 6, o qual adota a arquitetura e o modelo de dados apresentados no Capítulo 5. Todavia, a constatação da eficácia deste mecanismo só pôde ser realizada através da construção de um protótipo básico. Este protótipo é discutido neste capítulo, sendo apresentado em partes nas próximas seções. A primeira seção discute o processo de escolha do *groupware* utilizado como base neste protótipo. A seguinte apresenta uma visão mais detalhada deste *groupware* para, logo após, discutir o processo de adaptação propriamente dito, incluindo aí detalhes importantes do processo de implementação do *framework* BW. Por fim, é apresentado o processo de construção da interface com o usuário e as considerações finais relativas ao processo de construção do protótipo completo.

7.1 Escolha do *Groupware*

Para a construção do primeiro protótipo envolvendo o *framework* BW priorizou-se inicialmente a escolha de uma ferramenta de *groupware* simples, de fácil compreensão e manipulação. Isto pois, ao se trabalhar com uma ferramenta *groupware* simples, poder-se-ia realmente avaliar a complexidade de manipulação do próprio *framework* BW, já que o *groupware* em si não acrescentaria um grau elevado de dificuldade no processo de inclusão deste *framework*. Outra prioridade na escolha do *groupware* era a presença de código aberto ou passível de ser obtido. Esta disponibilidade é obrigatória, pois a inclusão do *framework* BW exige alterações neste código, principalmente com a inclusão das chamadas nos momentos adequados.

Mediante estas exigências, a escolha do *groupware* recaiu sobre a ferramenta COPSE/CUTE, apresentada em linhas gerais em meio ao Capítulo 3. O COPSE foi desenvolvido dentro da dissertação de mestrado de Márcio Dias, junto à COPPE/UFRJ [DIA 98] e se apresentou como a melhor alternativa de ambiente para testar as idéias desta dissertação. Trata-se de uma ferramenta de *groupware* simples, de fácil manuseio e com uma ótima documentação, tanto através da própria dissertação de Dias [DIA 98], quanto através de comentários incluídos em seu código fonte, facilitando muito a sua compreensão e manipulação. Além disso, a estreita cooperação entre o grupo desenvolvedor do COPSE, junto à UFRJ, e o grupo desenvolvedor deste trabalho, junto à UFRGS, tornou possível a obtenção de seu código fonte e também incentivou o desenvolvimento deste protótipo com o COPSE/CUTE, uma vez ser a percepção de eventos no passado uma das carências desta ferramenta de *groupware*.

Outra vantagem apresentada pelo COPSE/CUTE é o fato deste ter sido construído inteiramente em Linguagem Java, a mesma escolhida para o desenvolvimento do *framework* BW. A escolha pelo Java, ao invés de uma linguagem mais tradicional como C/C++, para uma primeira implementação deste trabalho vem de algumas características

marcantes do Java. Este tem se mostrado uma linguagem segura, possuindo internamente diversas restrições neste sentido e tendo ainda tratamento de exceções nativo. O Java também tem se mostrado uma linguagem bastante simples, de fácil programação, facilitando o desenvolvimento rápido de aplicações completas. Por tudo isso, o Java foi escolhido para a construção da primeira versão do *framework* BW, tornando o COPSE/CUTE uma escolha natural para este primeiro protótipo.

O COPSE (*Collaborative Project Support Environment*) é formado por um ambiente para o desenvolvimento cooperativo de *software* e por um *framework* para a construção de aplicações *groupware*, acopladas ou não a este ambiente, com o mesmo objetivo. O CUTE (*Collaborative UML Technique Editor*) é uma destas ferramentas construídas sobre o *framework* do COPSE. Trata-se de um editor cooperativo para a modelagem de *softwares* orientados a objetos, focando as atividades de diagramação deste, permitindo a edição de diagramas de classes simples, usando elementos de classes e de relacionamento, com notação UML [DIA 98], [DIA 97].

Embora sejam essencialmente voltados para o ambiente síncrono, tanto o *framework* COPSE quanto o editor CUTE (referenciados aqui pela sigla COPSE/CUTE), são o alvo principal na construção deste protótipo, sendo neles incluído o *framework* BW. O motivo para esta inclusão vem da própria natureza da atividade envolvida dentro do COPSE: o desenvolvimento cooperativo de *software*. Esta não é uma atividade passível de ser concluída em uma única sessão, necessitando normalmente de várias reuniões do grupo para que seu objetivo seja atingido. Todavia, o COPSE/CUTE não oferece um mecanismo de suporte à percepção de eventos no passado, que permita aos membros do grupo relembrarem, ou até mesmo se inteirarem, do que foi realizado nas últimas reuniões do grupo, prejudicando o andamento do trabalho à medida que o número de sessões necessárias aumenta. Com a inclusão do *framework* BW, este problema é minimizado e as atividades do grupo podem transcorrer normalmente, mesmo que divididas entre várias reuniões.

Assim, a ferramenta de *groupware* COPSE/CUTE mostrou-se ideal como ambiente de teste para este protótipo, já que, além de necessitar de um mecanismo de suporte a *awareness* de eventos no passado, ela também é uma ferramenta simples e tem seu código fonte disponível e acessível. Dessa forma, fez-se um estudo detalhado deste *groupware*, a fim de identificar suas características mais marcantes e, principalmente, identificar os pontos dentro de seu código fonte onde deveriam ser incluídas as chamadas ao *framework* BW. O resultado deste estudo é resumido na próxima seção.

7.2 Estrutura do COPSE/CUTE

Como já foi mencionado anteriormente, o CUTE é um editor cooperativo de diagramas de classes UML construído sobre o *framework* COPSE, referenciados como um todo neste capítulo pela sigla COPSE/CUTE. O COPSE/CUTE foi todo construído utilizando a linguagem Java 1.1, sendo todo organizado em pacotes bem definidos, semelhante ao próprio *framework* BW, que foi dividido em quatro pacotes, conforme foi discutido no capítulo anterior. A Fig. 7.1 a seguir esquematiza a organização dos pacotes no COPSE/CUTE.

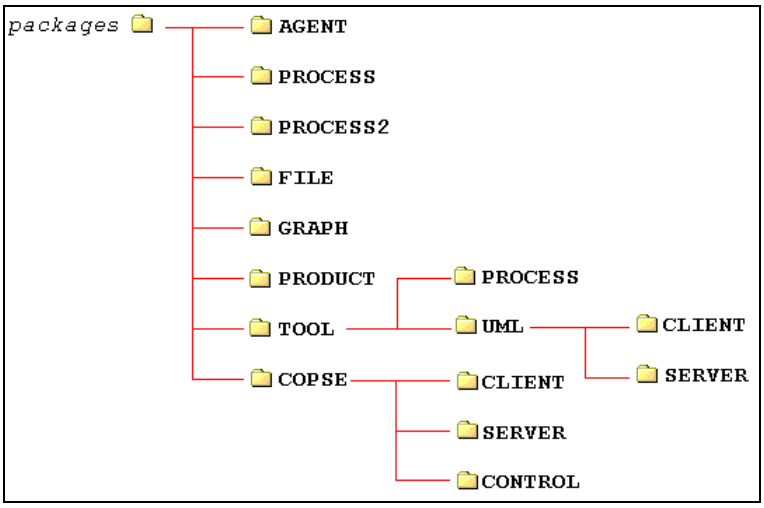


FIGURA 7.1 - Pacotes do COPSE/CUTE

A maioria destes pacotes constitue a base do ambiente de desenvolvimento cooperativo de *software*, embora seja possível destacar dois pacotes de maior interesse para este trabalho: o pacote “COPSE” e pacote “TOOL”. O primeiro traz as principais classes que compõem o *framework* COPSE, enquanto o segundo traz algumas ferramentas desenvolvidas com os demais pacotes, incluindo o CUTE, colocado no pacote “TOOL.UML”. Este é dividido em dois outros pacotes, chamados “client” e “server”, os quais refletem a arquitetura cliente/servidor adotada pelo COPSE/CUTE.

O COPSE/CUTE adota uma arquitetura cliente/servidor, com um protocolo de comunicação TCP/IP baseado no envio de objetos inteiros serializados. Estes objetos são gerenciados e armazenados pelo servidor, deixando para cada cliente apenas a responsabilidade da apresentação visual, baseada em cópias consistentes destes objetos. Cada servidor trabalha sobre um único projeto, ou seja, sobre um único diagrama de classes, e é capaz de atender a diversos clientes sobre este projeto. Isto significa que todos os clientes que estiverem conectados a um mesmo servidor estarão obrigatoriamente trabalhando sobre um único diagrama de classes, uma vez que o servidor não é capaz de trabalhar simultaneamente com mais de um diagrama. Da mesma forma, cada cliente é capaz de conectar-se com um único servidor por vez, operando sobre somente um diagrama de classes em um dado momento.

Tanto o lado cliente quanto o lado servidor possuem uma base comum formada por quatro gerentes principais: um gerente de sessão, que controla as conexões estabelecidas e as informações sobre os membros; um gerente de dados, responsável pelo controle de concorrência; um gerente de cooperação, responsável pelos elementos de controle que fornecem os mecanismos de cooperação e um gerente de eventos, que distribui e trata os eventos.

O gerente de sessão é composto por sessões de ferramenta (“CToolSession”), criadas para cada ferramenta acionada. Esta sessão de ferramenta possui informações sobre o tipo de aplicação, sobre o contexto em que foi acionada e mantém as sessões dos usuários (“CUserSession”) ativas. Cada aplicação possui também um gerente de dados (“CFileController”), que é responsável pelo controle de concorrência. Este controle de concorrência é feito através de um mapeamento de todos os elementos (“CElement”) que constituem um arquivo (“CFile”) em controladores de concorrência do elemento (“CElementController”). Cada aplicação também possui um manipulador de eventos (“CEventHandler”), responsável por receber e identificar um evento enviado

pela conexão cliente-servidor. Identificado este evento, o manipulador decide que operação realizar, de acordo com a situação. Um manipulador especializado para a aplicação cliente (“CClientEventHandler”) irá somente realizar as operações referentes ao evento, enquanto o manipulador do servidor (“CServerEventHandler”) irá também retransmitir os eventos aos clientes conectados, caso a operação decorrente dele seja bem sucedida. Além disso, cada aplicação cliente ou servidora possui somente um gerente de cooperação (“CCollaborationManager”), responsável por gerenciar os mecanismos de colaboração ativos, como o mecanismo de comunicação síncrona e a percepção dos colegas ativos no momento. Cada um destes mecanismos é manipulado por um manipulador específico (“CCollaborationHandler”), que trata os seus eventos.

A Fig. 7.2 a seguir traz um trecho do modelo de classes do servidor, onde é possível observar estes quatro gerentes. Também é possível observar outras classes, voltadas para conexão com o cliente, como é o caso da classe "ServerSocket". Esta classe é responsável por atender os pedidos de conexão do cliente e dá início ao processo de estabelecimento desta conexão, durante o qual são recebidas informações sobre o cliente e o usuário. Uma vez aceita esta conexão, um novo objeto "CUserConnectionManager" é criado para atender toda a comunicação entre o novo cliente e o servidor, incluindo os eventos enviados pelo cliente para o servidor, os quais são repassados para o manipulador de eventos ("CServerEventHandler").

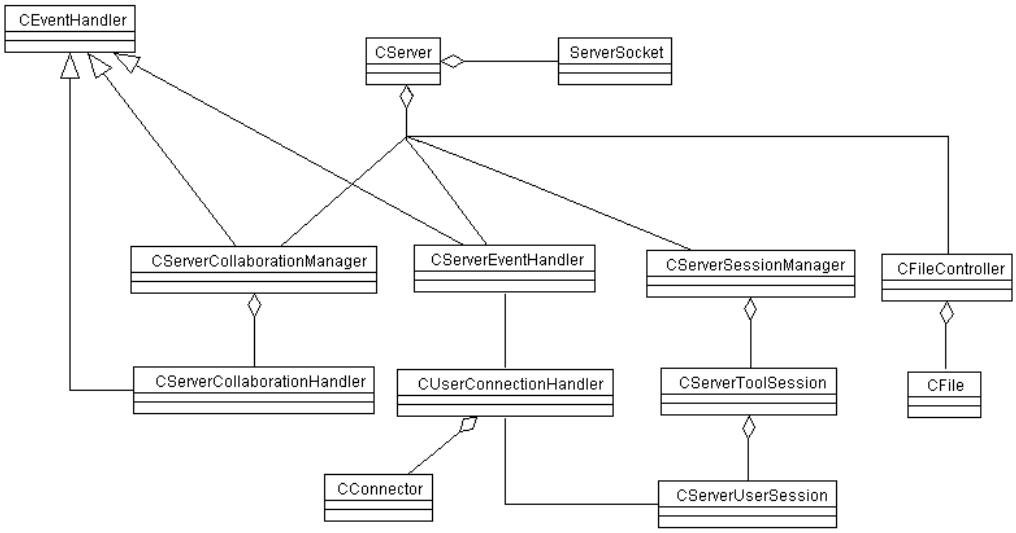


FIGURA 7.2 - Diagrama de classes do servidor COPSE/CUTE

No lado do cliente, uma estrutura semelhante pode ser observada, como mostra o trecho do diagrama de classes apresentado na Fig. 7.3 abaixo. Nele, os mesmos gerentes estão presentes, mas agora desempenhando funções específicas para o cliente. Pode-se observar na Fig. 7.3 os gerentes de sessão ("CClientSessionManager") e de dados ("CFileController"), os quais mantém atualizados, consistentes e disponíveis todas as informações sobre a sessão e o diagrama que está sendo manipulado no momento, enquanto que o gerente de cooperação ("CClientCollaborationManager"), também observado na Fig. 7.3, mantém informações sobre os recursos de cooperação presentes e notifica a interface com o usuário quando um de seus manipuladores recebe um evento. Estes eventos também podem ser recebidos através da conexão entre cliente e servidor, feita com auxílio de um conector específico ("CEventConnector")

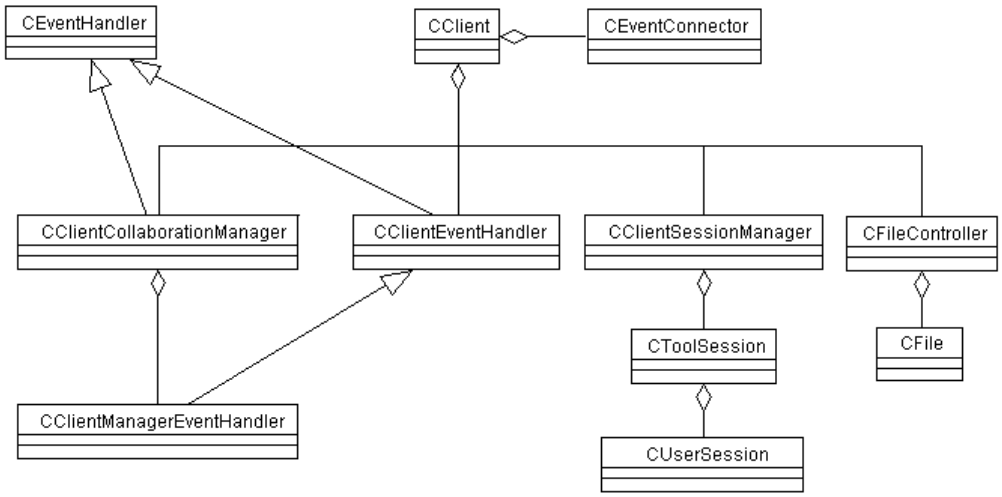


FIGURA 7.3 - Diagrama de classes do cliente COPSE/CUTE

Além da estrutura apresentada na Fig. 7.3, o cliente também possui em sua arquitetura uma camada dedicada à interface com o usuário, a qual se comunica com a apresentada na figura acima, chamada de camada de gerência, através de "*Listeners*", implementando o *design pattern Observer*. Através destes *listeners*, a camada de gerência notifica a interface sobre os eventos ocorridos. Toda esta interface é controlada por um gerente central ("CGUIManager"), que utiliza os serviços da camada de gerência através da classe "CClient". Este gerente central conta com a ajuda de dois auxiliares, um para a colaboração ("CGUICollaborationManager") e um específico para a ferramenta ("CGUIToolManager"), os quais representam os *listeners*. Eles organizam e delegam suas responsabilidades aos controles de interface ("CGUIControl"), que administram, cada um, um componente específico da interface com o usuário.

Logo, os eventos são propagados da seguinte forma: assim que são gerados os eventos junto à interface de um cliente, os controladores dos elementos de interface deste informam os seus gerentes (de colaboração ou/e ferramenta) sobre os eventos gerados. Estes eventos são repassados, então, da camada de interface do cliente para a sua camada de gerência, que, por sua vez, os repassa ao servidor. Este recebe os eventos, processa-os e os repassa aos demais clientes, que recebem estes eventos pela camada de gerência, a qual notifica a camada de interface através dos *listeners.*

Outro ponto interessante a ser observado é quanto ao modelo hierárquico. O COPSE/CUTE não adota um modelo concreto. Qualquer usuário, sem a necessidade de qualquer autenticação, pode se conectar a um servidor e participar da edição do diagrama de classes fornecido pelo servidor. Esta ausência se deve ao fato da ferramenta CUTE, construída com o *framework* COPSE, não ter sido integrada ao ambiente de desenvolvimento, o qual deveria manter não só as restrições de segurança, mas também de processo (modelo cognitivo) e do modelo hierárquico. Assim, nenhuma restrição de segurança sobre o diagrama de classes sendo editado é realizada, e todos os usuários conectados a um mesmo servidor poderão alterar o diagrama, gozando de uma igualdade completa de direitos.

7.3 Processo de Adaptação

O processo de adaptação do *framework* BW ao *groupware* COPSE/CUTE envolveu primeiramente a implementação do próprio *framework* BW em linguagem Java. Esta implementação foi realizada sobre ambiente Linux, usando a versão 1.1 do JDK _ *Java Development Kit*, e seguiu o modelo apresentado no capítulo anterior. Foram implementados os quatro pacotes definidos anteriormente (BW.kernel, BW.storage, BW.control, BW.interface), observando a seguinte nomenclatura: todas as classes pertencentes ao pacote *kernel* começam pela seqüência "BW_", as do pacote *control*, pela seqüência "CL_", as do pacote *storage*, por "ST_", as de interface, pelas iniciais "UI_", e, por fim, as classes de fachada são identificadas pela seqüência "FC_", antes das iniciais de seu pacote, conforme resume a Fig. 7.4 abaixo. Esta nomenclatura permite uma identificação rápida do pacote ao qual pertence uma determinada classe e, com isso, dá uma idéia básica da funcionalidade da classe.

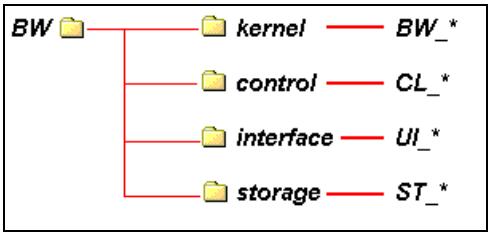


FIGURA 7.4 - Estrutura de pacotes do framework BW

Realizada a implementação do *framework* BW, passou-se, em seguida, para o processo de adaptação deste *framework* ao *groupware* COPSE/CUTE propriamente dito. Este processo teve seu início na implementação da base de informações, seguindo o modelo ER definido no Capítulo 5. Optou-se para a implementação desta base pelo uso do banco de dados PostgreSQL. Esta escolha baseou-se em uma série de características deste SGBD favoráveis à implementação deste protótipo. A primeira delas é ser confiável. O PostgreSQL consiste em um SGBD objeto-relacional completo, o qual mantém os princípios básicos da atomicidade, consistência, integridade e durabilidade necessários à manutenção confiável dos dados gerados pelo *framework* BW. Além disso, o PostgreSQL pode ser integrado facilmente com aplicações Java, através de seu *driver* JDBC _ *Java DataBase Connectivity*, o qual permite que aplicações Java possam acessar e manipular uma base de dados PostgreSQL.

Além disso, o PostgreSQL possui um servidor próprio, que permite o uso de conexão de rede para a manipulação da base de dados. Esta é uma excelente vantagem para este trabalho, pois permite uma centralização da base de informações. Com o uso deste servidor do PostgreSQL, chamado de *postmaster*, é possível permitir que a base de dados fique localizada em um servidor central, sem a necessidade de se manter uma instância da camada de armazenamento junto a este servidor, pois cada instância, localizada junto a cada participante do grupo, poderá enviar diretamente suas informações para o *postmaster* através de uma conexão de rede TCP/IP convencional, como mostra a Fig. 7.5. Outra vantagem da adoção do modelo cliente/servidor pela camada de armazenamento é o fato deste ser o mesmo modelo adotado pelo COPSE/CUTE. Com tudo isto, a construção do protótipo torna-se mais simples, uma vez que o processo de inclusão do *framework* BW fica então limitado principalmente ao cliente COPSE/CUTE, não afetando demasiadamente o servidor deste ambiente e ainda respeitando a arquitetura adotada pelo COPSE/CUTE.

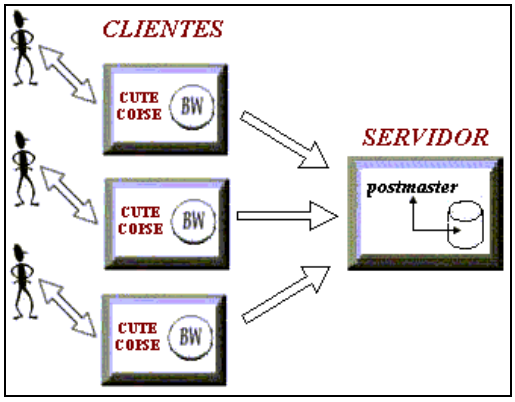


FIGURA 7.5 - Implementação centralizada da camada de armazenamento com o servidor *postmaster*

Feita a escolha pelo SGDB PostgreSQL, foi construída a base de dados a ser utilizada. Esta base construída seguiu o modelo de dados apresentado no Capítulo 5, com exceção apenas da questão do grupo, que seguiu a simplificação utilizada no *framework* BW, de permitir a existência de somente um grupo. O resultado dessa simplificação pode ser observado no diagrama entidade-relacionamento apresentado na Fig. 7.6 a seguir. Nesta figura, é possível se observar a presença das entidades "registro" (T_Register), "evento" (T_Event), "grupo" (T_Group), "participante" (T_Member), "papel" (T_Paper) e "*profile*" (T_Profile), e seus relacionamentos, semelhante ao que foi colocado no Capítulo 5 (ver Fig. 5.11). As exceções são os relacionamentos entre as entidades "participante", "grupo" e "papel", onde se manteve a mesma simplificação realizada no *framework* BW, de suportar apenas um único grupo. Com isso, eliminou-se a relação ternária entre estas entidades e manteve-se apenas relacionamentos binários, com cardinalidade 1:N, entre a entidade "grupo" (T_Group) e as entidades "participante" (T_Member) e "papel" (T_Paper).

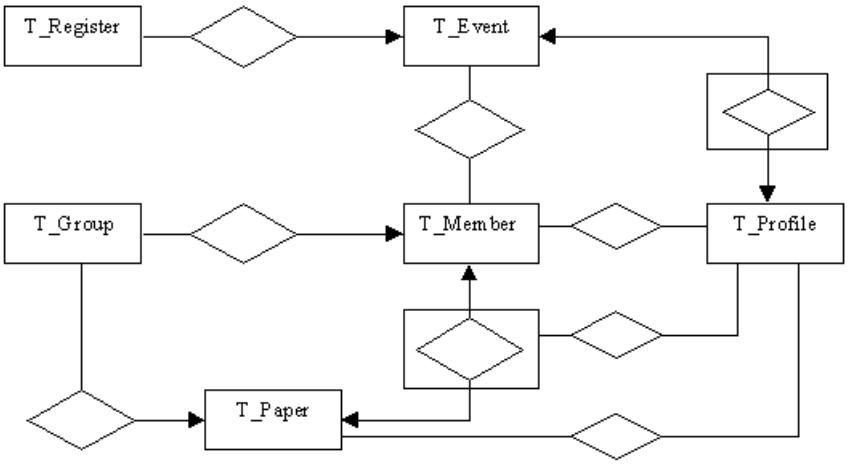


FIGURA 7.6 - Base de dados implementada sob o PostgreSQL

Concluída a construção da base de dados, passou-se para a especialização da classe "ST_Implementor", para manipulação da base de dados construída sobre PostgreSQL. A nova classe criada, chamada "ST_Database", foi desenvolvida especificamente para este protótipo. Ela mantém todas as informações sobre a estrutura de cada entidade da base de dados utilizada para armazenar o conteúdo das classes do

pacote *kernel* e ainda detém informações sobre como acessar e manipular esta base através do *driver* JDBC para o PostgreSQL. Esta classe "ST_Database", manipulada dentro do *framework* BW através de polimorfismo, por ser uma subclasse de "ST_Implementor", é a real responsável por enviar as informações geradas durante o funcionamento do mecanismo de contextualização para o servidor *postmaster*, que as armazena na base PostgreSQL.

Além da especialização da classe “ST_Implementor” para o caso específico deste protótipo e a implantação da base de dados, o processo de adaptação do *framework* BW para o *groupware* CUTE/COPSE envolveu também a criação de uma classe mediadora entre este *groupware* e o *framework* BW. Esta classe, chamada de "BW_Mediator", encapsula todas as chamadas necessárias ao *framework* BW e é somente através dela que o COPSE/CUTE envia suas requisições a este último. Estas requisições são realizadas essencialmente através de métodos de classe, o que dispensa a instanciação de vários objetos no decorrer do código do COPSE/CUTE, tornando as chamadas mais simples. Estas chamadas são repassadas internamente pelo mediador para uma única instância concreta sua, chamada "UML_BW_Mediator", definida especialmente para este protótipo. Esta instância detém as referências a todas especializações do *framework* BW, como a referência à "ST_Database", subclasse de "ST_Implementor", e às classes de interface com o usuário, que serão discutidas na próxima seção. Estas referências são passadas ao *framework* BW para sua utilização via polimorfismo no momento de inicialização deste, também feita pelo mediador concreto "UML_BW_Mediator". A Fig. 7.7 abaixo esquematiza esta estrutura, mostrando o mediador "BW_Mediator", especializado pela classe "UML_BW_Mediator", recebendo todas as chamadas vindas do COPSE/CUTE e as repassando para o BW.

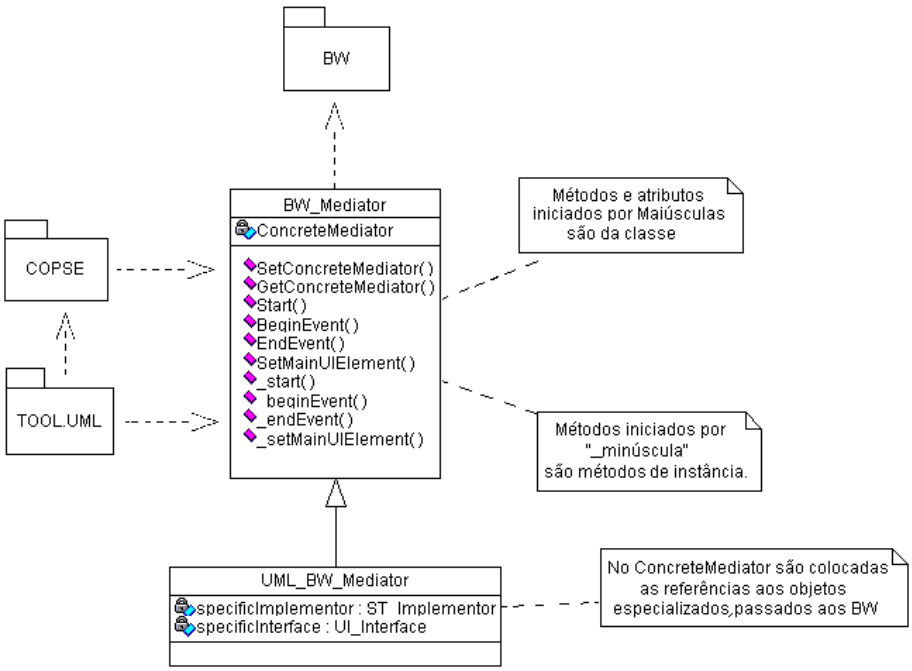


FIGURA 7.7 - Estrutura do mediador entre o COPSE/CUTE e *framework* BW

Através do uso deste mediador, introduziu-se no COPSE/CUTE chamadas a somente esta classe. Este mediador, através do uso de métodos estáticos, comporta-se como uma instância única., a qual repassa estas chamadas para os métodos de instância equivalentes da instância concreta que mantém internamente, representada neste protótipo por uma instância da subclasse "UML_BW_Mediator". Esta última é quem finalmente faz as devidas chamadas ao *framework* BW. Tudo isso objetiva simplificar as alterações realizadas sobre o COPSE/CUTE, reduzindo os possíveis impactos negativos destas alterações, como uma menor legibilidade e um maior acoplamento.

A criação desta classe mediadora é, na verdade, uma implementação do *design pattern Mediator,* em combinação com o *pattern Singleton.* O *pattern Mediator* propõe encapsular em um objeto a forma como um conjunto de outros objetos interagem., enquanto o *pattern Singleton* garante que a classe tenha somente uma instância. A idéia por trás do *design pattern Mediator* é introduzir um intermediário, o qual previne que os demais objetos em um grupo precisem referenciar uns aos outros explicitamente, reduzindo o número de interconexões. Já o *pattern Singleton* objetiva justamente manter uma única instância da classe, que deve ser acessível a todos os clientes através de pontos de acesso bem conhecidos [GAM 94]. A Fig. 7.8 e a Fig. 7.9 abaixo, ambas baseadas em Gamma et al. [GAM 94], trazem a estrutura dos *patterns Mediator* e *Singleton,* respectivamente. O uso destes *design patterns* simplificou bastante a comunicação entre o COPSE/CUTE e o *framework* BW, principalmente, sem sacrificar a estrutura independente do próprio *framework* COPSE.

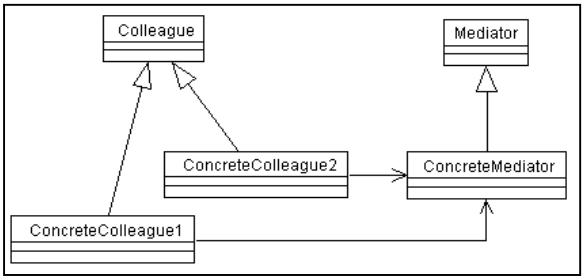


FIGURA 7.8 - Estrutura do *design pattern Mediator*

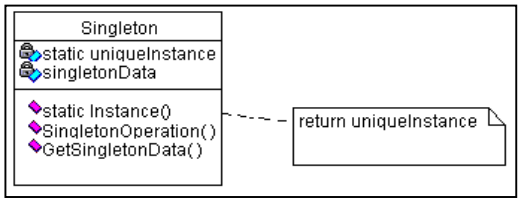


FIGURA 7.9 - Estrutura do *design pattern Sinaleton*

Detalhando um pouco mais o mediador específico criado para este protótipo, este não só é responsável pelo correto redirecionamento das chamadas feitas pelo COPSE/CUTE para o *framework* BW, mas também é o responsável pelo processo de inicialização das classes específicas deste protótipo, em especial a "ST_Database", para acesso à base PostgreSQL, e as classes de interface com o usuário. Além disso, o mediador específico também é o responsável pela inicialização das classes do próprio *framework* BW. Esta inicialização é feita com a ajuda de três outras classes, do próprio *framework* BW, construídas especialmente para esta finalidade. Estas classes, chamadas "ST_Startup", "CL_Startup" e "UI_Startup", foram incluídas dentro de cada um dos pacotes componentes do BW (*Storage, Control* e *Interface* respectivamente) e são responsáveis por realizar todo o processo de inicialização e organização das referências necessárias dentro do pacote. Desta forma, o mediador específico pode inicializar o *framework* BW, através destas classes de *startup,* sem, com isso, ferir o encapsulamento dos pacotes *Control, Storage* e *Interface,* como mostra o diagrama de seqüência da Fig. 7.10 a seguir, onde é possível se observar o mediador "BW_Mediator", usando os serviços do mediador concreto "UML_BW_Mediator", o qual chama seqüencialmente as três classes de *startup* assim que recebe o pedido de inicialização do sistema (método "Start").

Desenvolvido este mediador e as especializações necessárias, passou-se à introdução propriamente dita das chamadas à classe "BW_Mediator" dentro da estrutura do COPSE/CUTE. Estas chamadas foram incluídas em apenas algumas classes dos pacotes "COPSE" e "TOOL.UML" (em torno de 9% do total de classes foram afetadas), deixando as demais inalteradas. As classes afetadas são todas componentes do lado cliente da arquitetura COPSE/CUTE. Nenhuma classe do lado servidor foi afetada, pois os eventos no COPSE/CUTE são gerados durante a interação do usuário com o cliente e são, então, enviados para o servidor. Capturando estes eventos no lado do cliente, onde também as informações de *awareness* deverão ser apresentadas a este mesmo usuário, o servidor não precisou ser alterado. Mesmo assim, alterando-se somente o cliente, as informações resultantes das atividades de cada usuário puderam ser capturadas, enviadas e resgatadas pela camada de armazenamento do *framework* BW, através da classe "ST_Database", para a base de dados PostgresSQL centralizada.

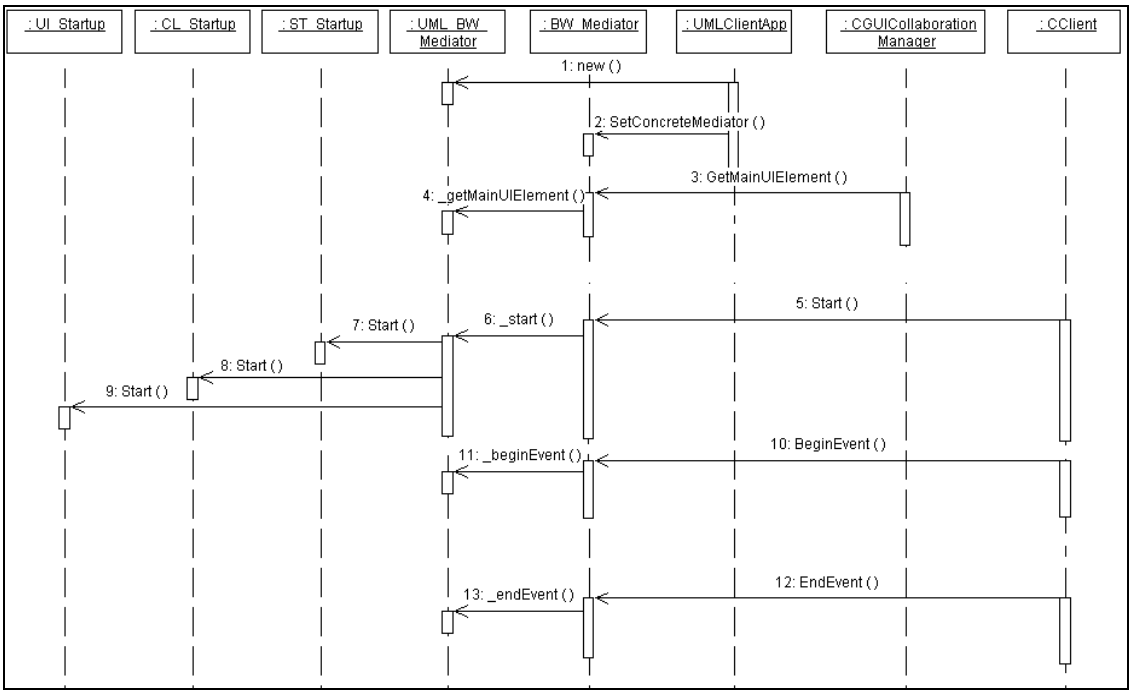


FIGURA 7.10 - Diagrama de seqüência para a inicalização do sistema

Estes pontos de alteração foram divididos em dois tipos: os de inicialização e os de monitoramento. Os de inicialização referem-se ao processo de *startup* do sistema e são essencialmente três: um na classe "UMLClientApp", outro na classe "CGUICollaborationManager" e um último junto à classe "CCLient". A primeira é responsável pela inicialização da aplicação CUTE e nela foi colocada a instanciação do mediador específico e a passagem desta referência para o mediador "BW_Mediator", através do método "SetConcreteMediator". A segunda pertence ao *framework* COPSE e é responsável pela montagem da interface com o usuário, e nela foi incluída a inicialização das classes de interface e a sua integração na interface do cliente, através de inserção de itens de menus. Enquanto a última, na classe "CClient", corresponde à ordem final de inicialização do *framework* BW, no momento em que o usuário entra no sistema. Todo este processo pode ser observado no diagrama de seqüência da Fig. 7.10

acima. Um diagrama de seqüência enfatiza o comportamento dos objetos em um sistema (representados por retângulos), mostrando as interações entre os objetos organizados em uma seqüência de tempo (representada pelos eixos pontilhados nos objetos) e de mensagens trocadas (setas rotuladas entre os objetos) [FUR 98]. Já os pontos de monitoramento foram distribuídos em oito classes distintas, estrategicamente colocadas no COPSE/CUTE, de acordo com os tipos de atividades para os quais se realizou o monitoramento. Estas classes variam entre classes de controle e de interface e foram escolhidas para tal, pois são capazes sozinhas de reunirem todas as informações necessárias para a construção dos eventos repassados ao *framework* BW, minimizando assim as alterações sobre o COPSE/CUTE.

Foram definidos nove tipos de atividades para serem monitoradas, consideradas importantes para o progresso do trabalho de edição do diagrama de classes entre uma seção e outra. Estas atividades consistem na edição, criação e remoção de classes e também de relacionamentos, no *logon* de usuários, no arquivamento do diagrama sendo editado ("*save"*), e no envio de mensagens de *broadcast* para o grupo. Estas atividades foram registradas junto ao *framework* BW por uma aplicação à parte do COPSE/CUTE, pois se optou dentro deste protótipo por manter um registro fixo e pré-determinado destes eventos, não havendo a necessidade de refazer este registro a cada execução do COPSE/CUTE. Esta aplicação à parte, chamada de "UML_Setup", faz o registro dos eventos junto ao *framework* BW, povoando a base de dados com as informações necessárias. Estas informações são resgatadas pelo *framework* BW durante a sua inicialização, fazendo com que não seja necessário um novo registro destas atividades a cada vez que um cliente ingressa no sistema.

Além do registro dos tipos de eventos a serem observados, esta aplicação "UML_Setup" também realiza a definição do *profile* por papel utilizado dentro do protótipo. Este *profile* é único para todo o sistema, pois, como já foi mencionado na seção anterior, o COPSE/CUTE não faz nenhum controle com relação aos usuários, permitindo que todos os participantes gozem dos mesmos direitos. Dessa forma, foi criado um único papel, chamado apenas de "participante", e um único *profile* para este papel. Este *profile* é criado uma única vez pela aplicação "UML_Setup" e contém os eventos de criação, remoção e edição de classes. Com isso, cada membro do grupo poderá ainda optar em seus *profiles* pessoais pelos eventos de criação, remoção e edição de relacionamentos, *logon* de usuários no sistema e mensagens de *broadcast* enviadas.

Analisando o que foi comentado anteriormente nesta seção, pode-se afirmar que o processo de adaptação do *framework* BW ao *groupware* COPSE/CUTE foi realizado em seis etapas, começando pela definição da base de dados e criação das classes específicas para a aplicação, passando pela análise da estrutura do *groupware* e criação do mediador específico, chegando à construção da aplicação "UML_Setup", para povoamento da base de dados, e concluindo com a inclusão das chamadas ao mediador dentro do *groupware*. Realizadas estas etapas, o próximo passo na construção deste protótipo consistiu na construção da interface com o usuário, seguindo o conjunto de orientações apresentado no capítulo anterior. Esta interface foi desenvolvida através da especialização das classes abstratas "UI_Element" e "UI_Event definidas no *framework* BW. A próxima seção descreve como foi realizado este processo da interface com o usuário, levando em consideração o que foi colocado nas orientações do Capítulo 6.

7.4 Construção da Interface

A construção da interface do mecanismo de contextualização representado pelo *framework* BW com o seu usuário partiu de um ponto único: a necessidade de se especializar as classes abstratas "UI_Element" e "UI_Event" definidas no pacote *Interface*, utilizando-se as orientações apresentadas no capítulo anterior. Analisando-se inicialmente estas classes, pôde-se observar que a sua implementação real apresenta uma peculiaridade. Estas "classes", na verdade, foram implementadas com o uso de um recurso especial do Java, as chamadas "interfaces". As interfaces Java nada mais são do que a descrição de classes abstratas, que, ao contrário das classes comuns Java, permitem o uso de herança múltipla, à medida que permite que uma classe concreta pode implementar mais de uma interface e, portanto, "herdar" os métodos e atributos definidos nestas interfaces. Dessa forma, a definição de "UI_Element" e "UL_Event" não representou um empecilho na implementação de suas "subclasses", uma vez que estas puderam ainda utilizar o mecanismo de herança tradicional do Java para especializar classes de interface já definidas pela API Java, como janelas e painéis.

Dessa forma, utilizando recursos de interface da própria API Java e implementando as interfaces "UI_Element" e "UL_Event", construiu-se a interface com o usuário, observando-se algumas das orientações para construção da interface colocada no capítulo anterior. A primeira destas orientações observadas foi a questão do uso da memória sem sobrecarregar o usuário. Buscou-se construir a interface de forma que as informações resgatadas do *framework* BW fossem apresentadas de forma resumida, buscando fornecer inicialmente apenas um *overview* do conjunto completo de informações. Buscou-se também chamar a atenção do usuário para estas informações, através da aplicação das orientações de uso de cores e gráficos. Foi desenvolvida, então, uma única implementação de "UI_Element", que constrói uma janela na qual é colocado um gráfico de barras colorido com a quantidade de ocorrências dos eventos de interesse do usuário, organizados por tipo. Esta janela foi desenvolvida de modo ainda a ser integrada facilmente à interface do CUTE, seguindo a orientação referente à integração entre *awareness* e *groupware*. A interface do CUTE já conta com menus, através dos quais o usuário solicita os recursos desejados, colocados, em sua maioria, em janelas à parte. A interface do *framework* BW foi, então, colocada da mesma forma, procurando manter a consistência com a interface do *groupware* (outra sugestão apresentada no Capítulo 6): chamada a partir de um item do menu "*Awarenes*" do CUTE e, quando acionada, colocada em uma janela à parte. Dessa forma, pôde-se integrar a interface do *framework* BW a do CUTE sem que o usuário sinta diferença entre esta e a interface dos demais recursos do COPSE. A Fig. 7.11 a seguir traz um exemplo desta janela desenvolvida, já integrada ao CUTE.

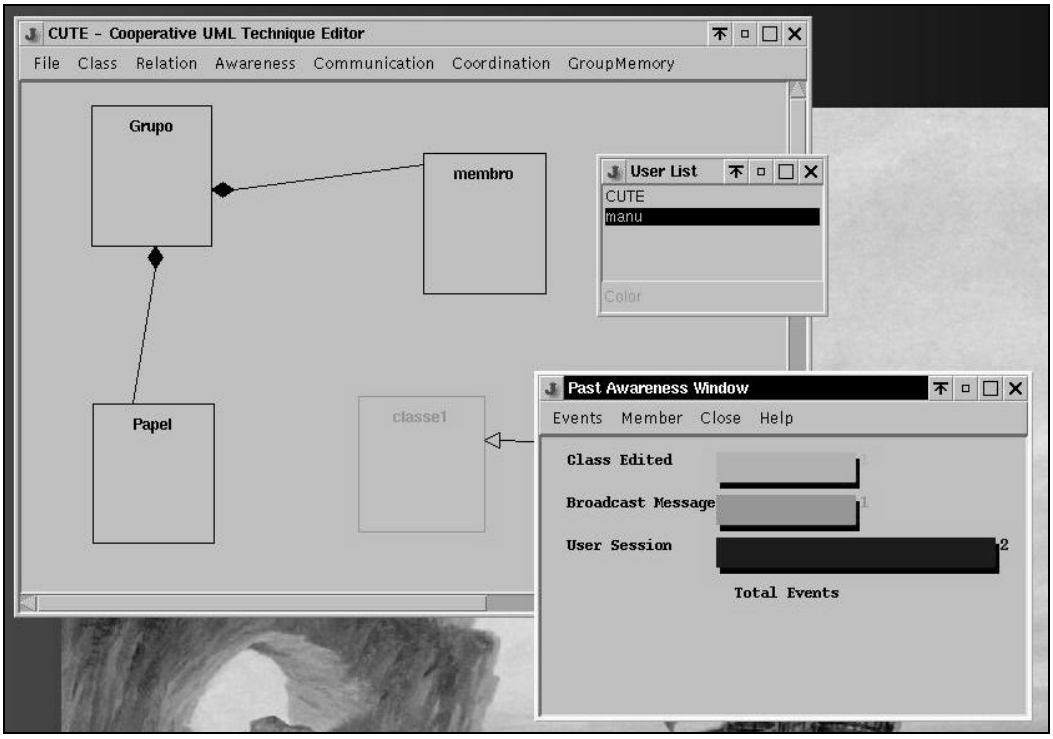


FIGURA 7.11 - Janela principal da interface com o usuário do *framework* BW junto ao CUTE

Além desta visão geral apresentada, foi criada também uma implementação de "UI_Event" para a apresentação mais detalhada dos eventos por tipo. Esta visão detalhada somente é mostrada quando requisitada pelo usuário, através de um *click* na barra correspondente ao tipo desejado no gráfico apresentado na visão geral (implementação de "UI_Element"). Esta visão mais específica apresenta inicialmente uma lista resumida de todos os eventos daquele tipo ocorridos. Caso o usuário queira ver um evento com todos os seus detalhes, basta selecionar o evento que deseja, conforme mostra a Fig. 7.12 abaixo. Dessa forma, procurou-se evitar a sobrecarga de informações sobre o usuário, à medida que ele somente terá todos os detalhes se assim o desejar, e também procurou-se facilitar o uso desta interface através do uso do conhecido mecanismo de "*point and click*", já largamente difundido entre as interfaces gráficas hoje utilizadas, como a do sistema MS Windows© e Motif®. Todavia, também seguindo a orientação de facilitar o uso, foi incluída junto à janela de *overview* principal um menu com uma chamada a uma pequena guia de ajuda, com um resumo das funções para utilização dos recursos deste mecanismo de suporte a eventos no passado.

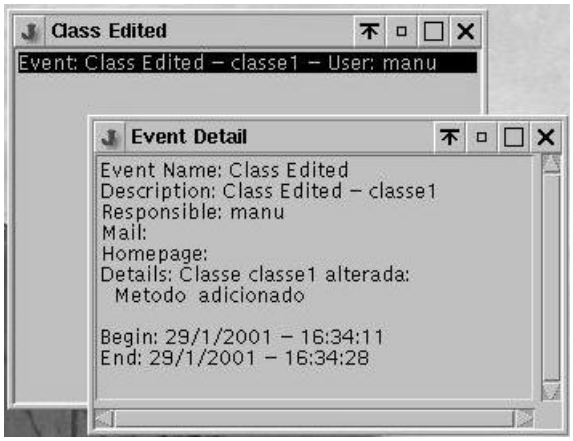


FIGURA 7.12 - Janelas com detalhes sobre os eventos ocorridos

7.5 Considerações Finais

Neste capítulo, apresentou-se de forma resumida o processo de construção do primeiro protótipo do *framework* BW, utilizando-se o *groupware* COPSE/CUTE. Todo este processo de construção do protótipo trouxe à tona algumas observações importantes. A primeira delas é quanto ao nível de conhecimento sobre o *groupware*. A introdução das chamadas ao *framework* BW dentro do *groupware* exige do desenvolvedor que as estiver executando um conhecimento profundo deste *groupware*. O desenvolvedor deve saber exatamente os pontos onde as informações desejadas estarão disponíveis, além de como incluir estas chamadas com o menor impacto possível sobre o código do *groupware*. No caso específico deste protótipo construído, este conhecimento necessário foi obtido através da cooperação com o grupo carioca responsável pelo desenvolvimento original do COPSE/CUTE. Sem este apoio de seus desenvolvedores originais, a construção deste protótipo não teria sido possível.

Outra consideração é com relação à criação das classes mediadoras. Estas classes intermediárias foram necessárias para diminuir o impacto da inclusão do *framework* BW sobre as classes pertencentes ao *framework* COPSE, pois este, em seu estado original, não possui dependências com relação à aplicação CUTE. Caso fossem inseridas chamadas diretas ao *framework* BW dentro do COPSE, estas estariam diretamente relacionadas a este protótipo com a aplicação CUTE, criando assim uma dependência, antes inexistente, entre o COPSE e a aplicação CUTE. Dessa forma, foi introduzida a figura do mediador genérico "BW_Mediator", o qual foi utilizado nas chamadas ao *framework* BW internas ao COPSE. A conseqüência desta inclusão foi a necessidade de mais outra especialização pelo desenvolvedor, a construção do mediador concreto, representado neste protótipo pela classe "UML_BW_Mediator".

Tratando especificamente sobre estas especializações, constatou-se que para a introdução do *framework* BW em uma ferramenta de *groupware* serão necessárias, no mínimo, quatro especializações: a da classe "ST_Implementor", para o acesso à base de dados real, a das classes de interface com o usuário, "UI_Element" e "UI_Event", e a do mediador. Isto faz com que seja necessário prover o desenvolvedor com informações suficientes sobre o próprio *framework* BW e sua estrutura, através de uma sólida documentação. Esta documentação inicialmente foi feita de três formas: através desta dissertação, do uso intenso de comentários junto ao código fonte e através da documentação automática da linguagem Java, feita com auxílio da ferramenta *javadoc*. Esta documentação é gerada em forma de páginas HTML sobre comentários especiais colocados dentro do código fonte. Todavia, durante a construção deste protótipo, observou-se que será necessária a construção de um manual específico para o processo de adaptação do *framework* BW ao *groupware*, um manual que aponte os passos a serem executados e os pontos a serem especializados, e que deverá ser construído posteriormente à finalização deste trabalho.

8 Conclusão

O presente trabalho teve seu início com uma revisão sobre a área de CSCW e um estudo aprofundado sobre percepção, culminando com sua classificação, apresentada no Capítulo 3. Este estudo permitiu a identificação de várias deficiências nas ferramentas de *groupware* hoje disponíveis, o que induziu a proposição de um mecanismo flexível para o suporte à percepção de eventos no passado.

A especificação deste mecanismo para o suporte à percepção de eventos no passado constitui a principal contribuição trazida por este trabalho. Sua proposta de flexibilidade e a possibilidade de inserção em mais de uma ferramenta de *groupware* constitui uma inovação na área, uma vez que, em sua imensa maioria, os mecanismos de suporte à percepção hoje disponíveis não abordam o suporte a eventos no passado e estão colocados sempre de forma casada a alguma ferramenta de *groupware* já existente. Este casamento entre *groupware* e suporte à percepção já foi apresentado como um problema por Sohlenkamp [SOH 98], o qual afirma que apesar de sua importância, o suporte sistemático a *awareness* em ferramentas de *groupware* é uma exceção, sendo que soluções *ad-hoc* ainda estão prevalecendo. Dessa forma, este trabalho representa um passo em direção a esta sistematização do suporte à percepção, no momento em que propôs e construiu um mecanismo independente com este objetivo.

Outra importante contribuição deste trabalho foi a proposta de uma classificação para o domínio da percepção. Esta classificação baseou-se em uma extensa pesquisa bibliográfica realizada na área e pode ser encarada como um resumo do que hoje existe em termos de suporte à percepção. Esta é uma classificação única em termos de abrangência, pois se buscou cobrir o maior número possível de mecanismos de suporte disponíveis, podendo vir a se tornar uma referência na área.

Além disso, a implementação do *framework* BW flexível o suficiente, a ponto de ser integrado a praticamente qualquer ferramenta de *groupware* que necessite de suporte à percepção de eventos no passado, constitui também outra importante contribuição deste trabalho. A inserção do *framework* BW no COPSE/CUTE, apresentada no Capítulo 7, demonstrou que a especificação deste *framework*, realizada no Capítulo 6, atingiu, de fato, o grau de flexibilidade desejado para este trabalho. Da forma como está implementado agora, o *framework* BW já pode ser inserido em outras ferramentas de *groupware*, mais complexas que o ambiente COPSE/CUTE, com a mesma necessidade de especialização de algumas classes, para sua melhor adaptação, conforme foi discutido no Capítulo 7.

A própria utilização do COPSE/CUTE para o primeiro protótipo do *framework* BW também constitui uma forma de contribuição deste trabalho. O uso do COPSE, na verdade, é a consolidação de um trabalho de intercâmbio entre UFRGS e UFRJ, que envolveu a realização, pela autora desta dissertação, de um estágio de aproximadamente três meses na UFRJ. Deste estágio, resultaram não só este protótipo, mas também alguns artigos (ver [PIN 00a] e [PIN 00b]) e a própria classificação para percepção, presente no Capítulo 3.

Também se pode citar, como contribuição importante deste trabalho, a própria abordagem direta e dedicada à questão dos eventos no passado. O suporte à percepção de eventos no passado, embora negligenciado por muitas ferramentas de *groupware*, constitui um aspecto importante para a cooperação. Graças a este suporte, situações muito comuns durante o trabalho em grupo podem ser visualizadas mais facilmente. Uma destas situações é a divisão em subtarefas menores de uma tarefa longa, a qual normalmente só seria contemplada por um suporte a eventos no passado contínuo. Com a inclusão do mecanismo para suporte à percepção de eventos somente no passado, à medida que estas subtarefas vão sendo concluídas, este mecanismo vai permitindo que as mesmas, concluídas, sejam observadas pelos demais colegas de grupo, sem a necessidade da presença também do suporte a eventos no passado contínuo. Com isso, viabiliza-se uma situação que, de fato, costuma ocorrer durante a cooperação, mesmo que face a face: à medida que os participantes vão concluindo trechos de suas atividades, vão apresentando seus resultados parciais aos colegas.

Outra situação comum durante o trabalho em equipe, viabilizada pelo suporte à percepção de eventos no passado, é a criação de rascunhos, mesmo quando a ferramenta de *groupware* não dispõe de uma área específica para isto. É muito comum que os participantes façam primeiro um rascunho de suas atividades, o qual, normalmente, não se deseja publicar aos colegas. Nesta situação, a restrição para a percepção de eventos no passado de somente utilizar atividades concluídas vem ao encontro dos interesses do participante, especialmente quando a ferramenta de *groupware* não dispõe de uma área dedicada para isso. Com esta restrição de somente atividades concluídas, somente quando o participante realmente dá uma atividade como concluída é que ela estará disponível para percepção dos colegas. Ou seja, os primeiros rascunhos desenvolvidos poderão ser trabalhados com calma e tranqüilidade e só publicados depois de consolidados.

Mesmo tendo atendido aos objetivos ao quais se propunha, muito ainda pode ser feito sobre este trabalho. O primeiro destes trabalhos futuros que pode ser citado aqui é a construção de um manual para auxiliar o processo de inserção do *framework* BW a uma ferramenta de *groupware* qualquer, conforme foi comentado no final do Capítulo 7.

Além disso, outro importante trabalho futuro a ser realizado é a inclusão do *framework* BW a uma ferramenta de *groupware* mais complexa que o COPSE/CUTE. O *groupware* escolhido para isto foi o editor cooperativo Byzance, rapidamente apresentado no Capítulo 3. Este processo de adaptação do *framework* BW ao Byzance vai, na verdade, fazer parte de um projeto maior, denominado "CEMT", de cooperação internacional, recentemente aprovado pelo CNPq.

Dentro desta idéia de seguir o desenvolvimento do *framework* BW, impulsionado pelas necessidades do projeto CEMT, outros trabalhos futuros ainda deverão ser realizados. Dentre eles, pode-se citar a implementação do *framework* BW também em linguagem C/C++, uma vez ser esta uma linguagem mais tradicional, utilizada por diversos sistemas de *groupware*, incluindo o próprio Byzance, além de possuir uma velocidade de execução significativamente maior que o Java, embora mais complexa. Esta nova implementação representará uma segunda opção de uso para o *framework* BW, ampliando suas possibilidades de uso.

Outro trabalho futuro a ser realizado sobre este é a expansão do mecanismo aqui especificado para o suporte também a eventos no passado contínuo, flexibilizando o mecanismo para também cobrir àquelas atividades que tiveram seu início em um momento no passado, mas ainda não foram concluídas. Além disso, pode-se ainda fazer

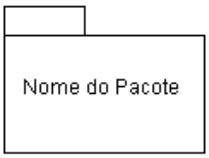
uma expansão do critério de caducidade hoje utilizado, para incluir relacionamentos entre eventos como critério de utilidade para uma informação. A mesma idéia de relacionamento entre eventos poderá também ser adotada junto ao esquema de *profiles* dos usuários, aumentando seu poder de expressão.

Por fim, outro trabalho futuro que pode ser citado aqui é a especificação de uma descrição genérica para modelos cognitivos com a finalidade de, futuramente, poder associar o mecanismo aqui desenvolvido a uma estrutura de *workflow,* que seria um último trabalho a ser realizado.

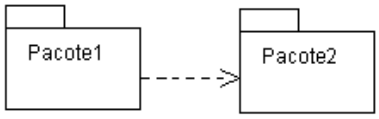
Anexo 1 Diagramas UML

Diagrama de Classes:

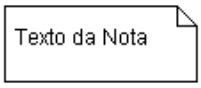
Pacote



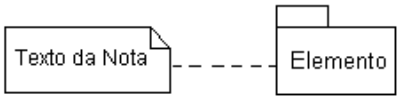
Dependência entre pacotes



Nota

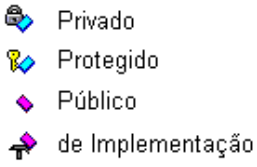


Ligação entre nota e elemento de referência

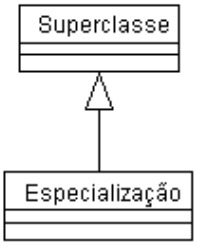


Classe

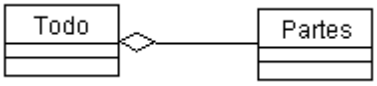
Visibilidade de atributos e métodos



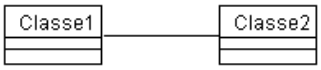
Especialização Generalização



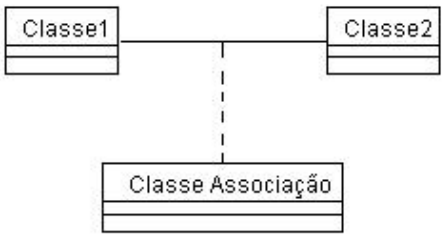
Agregação



Associação entre classes



Classe associação



Cardinalidade

0..0	Zero
0..1	De zero a 1
0..*	De zero a n
1..1	Um
1..*	Um a n
Mais de 1	

Navegabilidade

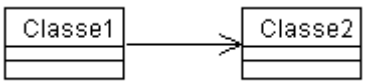
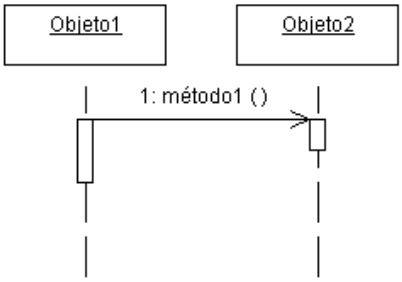


Diagrama de Seqüência

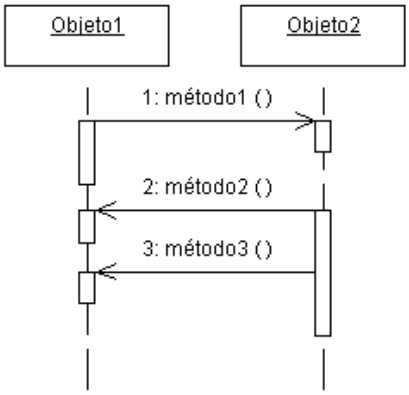
Objeto

Linha de tempo (em destaque, sob o objeto)

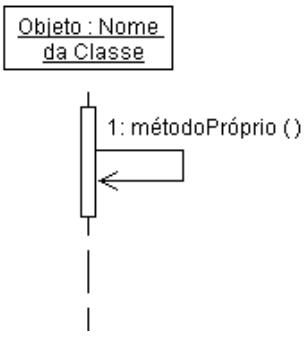
Chamada a um método de outro objeto



Chamadas de métodos sucessivas



Chamada a um método próprio



Bibliografia

[ACK 98] ACKERMAN, Marks; HALVERSON, Cristine. Considering an organizations memory. In: CSCW, 1998. Seattle, Washington, USA. **Proceedings...** Seattle: ACM Press, 1998. p. 39-48.

[ALL 83] ALLEN, J. F. Maintaining Knowledge about Temporal Intervals. **Communications of the ACM,** New York, v. 26, n. 11. p. 832-843, Nov., 1983.

[ANU 94] ANUPAM, V.; BAJA; C. L. Shastra: Multimedia Collaborative Design Environment. **IEEE Multimedia,** New York, p. 39 a 49, Summer, 1994.

[APP 99] APPLETON, B. **Patterns and Software**: Essential Concepts and terminology. Disponível em: <http://www.enteract.com/~bradapp/>. Acesso em: mai.1999.

[ARA 97] ARAÚJO, Renata M.; DIAS, Márcio de S.; BORGES, Marcos R. S. Suporte por Computador ao Desenvolvimento Cooperativo de Software: Classificação e Propostas. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 11., 1997. Fortaleza, CE. p. 299-314. **Anais…** Fortaleza:SBC, 1997.

[BAE 93] BAECKER, Ronald M.; NASTOS, Dimitrios; POSNER, Ilona R.; MAWBY, Kelly L. The user-centred iterative design of collaborative writing software. In: HUMAN FACTORS IN COMPUTING SYSTEMS - INTERCHI CONFERENCE, 1993. Amsterdan, The Netherlands. p. 399-405. **Proceedings…** Amsterdan, 1993.

[BEA 92] BEADOUIM-LAFON, Michel; KARSENTY, Alain. Transparency and awareness in a real-time groupware system. In: ANNUAL SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY – UIST, 1992. Montrey, CA. **Proceedings…** Montrey:ACM Press, 1992. p. 171-180

[BEL 95] BELLASSAI, Gerónimo; BORGES, Marcos R. S.; FULLER, David D.; PINO, José A.; SALGADO, Ana C. SISCO: A tool for improve meetings productivity. In: CRIWG, 1995. **Proceedings…** 1995. p.149-161.

[BEN 98] BENTLEY, Richard; HORSTMANN, Thilo; SIKKEL, Klass; TREVOR, Jonathan. **Suporting Cooperative Information Sharing with the World Wide Web:** The BSCW Shared Worksapce System. Disponível em: <http://www.w3.org/Conferences/WWW4/151/>. Acesso em: jul.1998.

[BOR 95] BORGES, Marcos R.S.; CAVALCANTI, Maria Cláudia R.; CAMPOS, M. L. M. Suporte por computador ao trabalho cooperativo. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA (JAI) - CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC), 15., 1995. Canela, RS. **Anais...** Canela: SBC, 1995.

[BOR 99a] BORGES, M.R.S.; PINO, J.A.; FULLER, D.; SALGADO, A. Key issues in the design of an asynchronous system to support meeting preparation, To appear In **Decision Support Systems**.

[BOR 99b] BORGES, M.R.S.; PINO, J.A. Awareness Mechanisms for Coordination in Asynchronous CSCW. In: WORKSHOP ON INFORMATION TECHNOLOGIES AND SYSTEMS – WITS, 9., 1999. Charlotte, North Carolina. **Proceedings...** Charlotte,1999.

[BRI 97] BRINKS, Tom; MCDANIEL, Susane. Small group discussions. In: WORKSHOP ON AWARENESS IN COLLABORATIVE SYSTEMS – CHI, 1997. **Proceedings…** Disponível em <http://www.usabilityfirst.com/groupware/awareness/workshop/small-groups.html>. Acesso em: out.1999.

[CAR 91] CARTER, D.; BACKER, B. S**. Concurrent Engineering: The Product Development Envieroment for the 1990s**. v. 1. San Jose: Mentor Graphics Cooporation, 1991.

[CAV 97] CAVALCANTI, M. Cláudia; BORGES, Marcos R. S., ENDO, Marcos Yuly. SISCO-RIO: Um sistema Assíncrono para Apoiar a Preparação de Reuniões. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 11., 1997. Fortaleza, CE. **Anais…** Fortaleza: SBC, 1997. p. 475-479.

[CHA 98] CHABERT, A. N.; GROSSMAN, E.; JACKSON, L.; PIETRONIEZ, S.; SEGUIN, C. Java Object Sharing in Habanero. **Communications of ACM,** New York, v.41, p. 69-76, June, 1998.

[CLA 98] CLAVELEIRA, C. **Les Applications de Travail Collaboratif**. Disponível em: <http://www.cru.fr/multimedia/applis_multi/applis_multi.fm.html>. Acesso em: jul.1998.

[COO 98] COOPER, J. Using Patterns design. **Communications of ACM,** New York, v.41, p. 65-68, June, 1998.

[CSC 98] CSCW and related issues. Disponível em: <http://dougal.derby.ac.uk/andreecscwandrel.html>. Acesso em: jul.1998.

[DAF 86] DAFT, R.L.; LENGEL, R.H. Organizational Information Requirements, Media Richness and Structural Design. **Management Science,** n.5, p. 554-571, May, 1986.

[DAT 91] DATE, Chris J. **Introdução a Sistemas de Bancos de Dados**. Rio de Janeiro: Campus, 1991.

[DAV 99] DAVIS, T. E. Clever Facade makes JDBC look easy. Cool Tools - **JavaWorld**, May, 1999. Disponível em: <http://www.javaworld.com/jw-05-1999/jw-05-cooltools.html>. Acesso em: mai.1999.

[DEC 98a] DECOUCHANT, Dominique; SALCEDO, Manuel Romero; QUINT, Vincent. **Structured Cooperative Authoring on the World-Wide Web.** Disponível em: <http://www.w3.org/pub/Conferences/WWW/Papers/91/>. Acesso em: set.1998.

[DEC 98b] DECOUCHANT, Dominique; SALCEDO, Manuel Romero; SERRANO, M. **Principes de Conception d'une Application d'Édition Coopérative de Documents sur Internet**. Disponível em <ftp://ftp.inrialpes.fr/pub/opera/publications/IHM97.ps.Z>. Acesso em: set.1998.

[DIA 97] DIAS, Márcio; XEXÉO, Geraldo. Editor Cooperativo para Diagramação de Software OO. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 11., 1997. Fortaleza, CE. **Anais...** Fortaleza: SBC, 1997. p. 499-502

[DIA 98] DIAS, Márcio de S**. COPSE**: Um ambiente de suporte ao projeto cooperativo de software. Rio de Janeiro, RJ: COPPE/UFRJ, 1998. Dissertação de mestrado.

[DIE 96] DIETRICH, Elton. **Projeto de um sistema de Suporte à Autoria Cooperativa de Hiperdocumentos**.. Porto Alegre: CPGCC/UFRGS, 1996. Dissertação de mestrado.

[DOM 98] DOMINGOS, Henrique J. L.; PREGUIÇA, Nuno M.; MARTINS, J.L. Coordenation and awareness support for adaptative CSCW sessions. In: International Workshop on Groupware – CRIWG, 4., 1998, Armação de Búzios, RJ. **Proceedings**… Armação de Búzios, 1998.

[DOU 97] DOURISH, Paul. Extending awareness beyond synchronous collaboration. In: WORKSHOP ON AWARENESS IN COLLABORATION SYSTEMS – POSITION PAPER - CHI, 1997. Atlanta, USA. **Proceedings**... . Disponível em: <http://www.best.com/~jpd/chi97-awareness.html>. Acesso em: jul.1999.

[ELL 91] ELLIS, C. A.; GIBBS, S.J.; REIN, G.L. Groupware: Some issues and experiences. **Communications of the ACM**, New York, v.34, n.1, p. 38-58. Jan. 1991.

[FER 98] FERNANDES, Jorge Henrique Cabral. CIBERESPAÇO: Modelos, Tecnologias, Aplicações Perspectivas. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA (JAI) - CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC), 17., 1998. Belo Horizonte, MG. **Anais...** Belo Horizonte: SBC- DCC/UFMG, 1998.

[FRI 97] FRIZKE, Udo Jr.; FARINES, Jean-Marie. Modelagem e Implementação de Aplicações de Computação Cooperativa em Ambientes Distribuídos. In: SEMINÁRIO INTEGRADO DE HARDWARE E SOFTWARE - CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO – SBC, 16., 1997. Brasília, 1997 **Anais...** Brasília: SBC - UnB, 1997.

[FUR 98] FURLAN, José Davi. **Modelagem de objetos através da UML – the Unified Modeling Language**: análise e desenho orientados a objeto. São Paulo : Makron Books, 1998.

[GAM 94] GAMMA, E.; HELM, R.; VLISSIDES, J.; JOHNSON, R. **Design patterns:** elements of reusable object-oriented software. Addison-Wesley, 1994.

[GMD 00a] GMD. **Basic Support for Cooperative Work**. Disponível em: <http://orgwis.gmd.de/projects/BSCW/>. Acesso em: mar.2000.

[GMD 00b] GMD. **BSCW**. Disponível em: <http://bscw.gmd.de/>. Acesso em: jan.2000.

[GRE 96] GREENBERG, Saul. Peepholes: Low cost awareness of one's community. In: SHORT PAPER - CHI, 1996. **Proceedings**... Disponível em: <http://www.acm.org/sgchi/chi96/proceedings/shortpap/Greenberg2/sg1txt.html>. Acesso em: jul.1999.

[GRE 98] GREENBERG, Saul; ROSEMAN, Mark. Groupware toolkits for synchronous work. In: BEAUDOUIN-LAFON, M. (ed.)**. Computer-Supported Cooperative Work:** trends in software series. John Wiley & Sons, 1998. p. 135-168. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: jan.2000.

[GRE 00] GREENBERG, Saul**. GroupLab:** Laboratory for Computer Supported Cooperative Work & Human Computer Interaction**.** Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/>. Acesso em: jan.2000.

[GRU 91] GRUDIN, J. CSCW: Introduction. **Communications of ACM**, New York, v.34, n.12, p. 30-34, dez., 1991.

[GRU 96] GRUDIN, Jonathan. CSCW, groupware and workflow: experiences, state of art, and future trends. In: CHI, 1996. Tutorial. **Proceedings**... Disponível em: <http://www.acm.org/sigchi/chi96/proceedings/tutorials/Grudin/jtg-txt.html>. Acesso em: out.1999.

[GRU 98] GRUDIN, J. Why CSCW applications fail: problems in design and evaluation of organizational interfaces. In: CONFERENCE ON CSCW, 2., 1998, Portland, Oregon, USA. **Proceedings...** Portland,1998.

[GUT 96] GUTWIN, Carl; ROSEMAN, Mark. A usability study of workspace awareness. In: SHORT PAPER – CHI, 1996. **Proceedings**... Disponível em: <http://www.acm.org/sigchi/chi96/proceedings/shortpap/Gutwin4/cg4txt.html>. Acesso em: out.1999.

[GUT 98] GUTWIN, Carl; GREENBERG, Saul. Design for individuals, design for groups: tradeoffs between power and workspace awareness. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE**,** 1998**. Proceedings...** Disponível em: <http://www.cpcs.ucalgary.ca/grouplab/papers/>. Acesso em: set.1999.

[GUT 99] GUTWIN, Carl; GREENBERG, Saul. **A framework of awareness for small groups in shared-workspace groupware**. Canada : Departament of Computer Science, University of Saskatchewan, 1999. (Technical Report 99-1). Disponível em: <http://www.cpcs.ucalgary.ca/papers/1999/99-AwarenessTheory/html/ theory-tr99-1.html>. Acesso em: set.1999.

[HAA 93] HAAKE, Anja; HAAKE, Jörg M. Tale CoVER: exploiting version support in cooperative systems. In: HUMAN FACTORS IN COMPUTING SYSTEMS - INTERCHI CONFERENCE, 1993, Amsterdan, The Netherlands. **Proceedings...** Amsterdan,1993. p. 406-412.

[HIX 93] HIX, Deborah; HARTSON, H. Rex. **Developing user interfaces**: ensuring usability through products & process**.** John Wiley & Sons, 1993.

[HOL 94] HOLLINSWORTH, David; WHARTON, Peter. An Architeture for Developing CSCW. In: SPURR, Kathy; LAYZELL, Paul; JENNISON, Leslie; RICHARDS, Neil. **Computer Support for Co-Operative Work. Computer Support for Co-Operative Work.** Chichester, EUA: John Wiley, 1994.

[HUD 96] HUDSON, Scott E.; SMITH, Ian. Techniques for fundamental privacy and disruption trade-off in awareness support systems. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERTATIVE WORK – CSCW, 1996, Boston, Massachusetts. **Proceedings**... Boston: ACM Press, 1996. p. 248 - 257.

[IET 98] INTERNET Engineering Task Force. Disponível em: <http://www.ietf.org/>. Acesso em: nov.1998.

[ISA 95] ISAACS, Ellen A.; MORRIS, Trevor; RODRIGUEZ, Thomas K.; TANG, John C. A comparation of face-to-face and distributed presentations. In: HUMAN FACTORS IN COMPUTING SYSTEMS – CHI, 1995, Denver, Colorado, USA. **Proceedings...** Denver, 1995. p. 354-361.

[JAC 96] JACOBS, Stephan; GEBHARDT, Michael; KETHERS, Stefanie; RZASA, Wojtek. Filling HTML Forms Simultaneously: CoWeb – Architecture and Funcionality. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE, 15., 1996, Paris, França. **Proceedings...** Paris, 1996.

[JEN 98] JENNIGS, Nicholas R.; WOOLDRIDGE, Michael J. **Agent Technology: fondations, applications and markets**. Springer Computer Science/UNICOM, 1998. Disponível em: <http://www.springer.de/comp/lncs/agent/index.html>. Acesso em: mai.2000.

[KLO 99] KLOECKNER, K. Computer Supported Cooperative Work (CSCW). In: Session 2 - Euromicro Workshop on Parallel and Distributed Processing, 7. **Proceedings…** Disponível em <http://www.computer.org/proccedings/euromicro-pdp/0059/00590120abs.htm>. Acesso em: out.1999.

[LEE 97] LEE, Alison. Awareness research based on Nynex Portholes. In: Workshop on Awareness in Collaborative Systems – CHI, 1997. **Proceedings...**. Disponível em: <http://www.cs.toronto.edu/~alee/chi97-awareness.html>. Acesso em: jul.1999.

[LEI 99] LEITE, Letícia Lopes; LIMA, José Valdeni de. Modelos de tempo para apresentações multimídia complexas. In: Semana Acadêmica do PPGC/UFRGS, 4., 1999, Porto Alegre, RS. **Anais…** Porto Alegre: PPGC/UFRGS, 1999. Disponível em: <http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana99/leticia/leticia.html>. Acesso em: out.2000.

[MAN 99] MANSFIELD, Tim; KAPLAN, Susan; FITZPATRICK, Geraldine; PHELPS, Ted; FITZPATRICK, Mark; TAYLOR, Richard; SEGALL, Bill; HERRING, Charles; JOHNSON, Philip; BERRY, Andrew. **Evolving Orbit:** a progress Report on Builds Locales. Disponível em: <http://www.dstc.edu.au/Research/Projects/Worlds/Publications/EnvolvingOrbit/index.html>. Acesso em: out.1999.

[MAR 97] MARIANI, John A. SISCO: Providing a cooperation filter for a shared information space. In: ACM SIGGROUP CONFERENCE ON SUPPORTING GROUPWORK – GROUP, 1997, Phoenix, Arizona. **Proceedings…** Phoenix: ACM Press, 1997.

[NOM 98] NOMURA, Takahiko; HAZAMA, Tan; HAYASHI, Koich; GUDMUNDSON, Stephan. Interlocus: workspace configuration mechanism for activity awareness. In: CSCW, 1998, Seattle, Washington. **Proceedings…** Seattle: ACM Press, 1998. p. 19-28.

[NUN 91] NUNAMAKER, J. F. Eletronic meeting systems to support group work. **Communications of the ACM**, New York, v.34, n.7, jul., 1991, p. 40 – 41

[OPE 99] OPERA GROUP. **Byzance**. Disponível em: <http://www.inrialpes.fr/opera/Byzance.fr.html>. Acesso em: jun.1999.

[ORT 99] ORTEGA, Chritian M.; FERNANDES, Clovis T. A framework for user-aware collaborative applications on the world wide web. In: WorkComp – ENCITA, 5., 1999, São José dos Campos, SP. **Anais...**.São José dos Campos:ITA, 1999.

[PAL 98] PALUDO, Marco; BERNETT, Robert; JAMHOUR, Edgard. Patterns in CSCW Modeling. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES, 12., 1998, Maringá, PR. **Anais...** Maringá: SBC, 1998. p. 37-52.

[PAR 99] PARSOWITH, Sara; FITZPATRICK, Geraldine; KAPLAN, Simon; SEGALL, Bill; BOOT, Jullian. **TickerTape:** notification and communication in a single line. Disponível em: <http://www.dstc.edu.au/Research/Projects/Worlds/Publications/apchi98.html>. Acesso em: out.1999.

[PIM 98] PIMENTEL, Maria da Graça; KUTOVA, Marcos A. S.; MACEDO, Alessandra A.; TEIXEIRA, Cesar A. C.. Hiperdocumentos Estruturados no Suporte ao Trabalho Cooperativo em Sistemas Abertos Distribuídos. In: SEMINÁRIO INTEGRADO DE SOFTWARE E HARDWARE - CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO – SBC, 25., 1998; Belo Horizonte, MG. **Anais...** Belo Horizonte: SBC-DCC/UFMG, 1998.

[PIN 99a] PINHEIRO, M. K. **Edição Cooperativa de Hiperdocumentos na WWW:** Trabalho individual. Porto Alegre:CPGCC/UFRGS, 1999. Disponível em: <http://www.inf.ufrgs.br/~manuele/mestrado.html>. Acesso em: dez.1999.

[PIN 99b] PINHEIRO, M. K.; LIMA, J. V. **Mecanismo de Monitoramento e Contextualização em Ambientes Cooperativos**. In: SIMPÓSIO NACIONAL DE INFORMÁTICA, 4., 1999, Santa Maria, RS. **Anais...** Santa Maria: CEUNIFRAN – UFSM, 1999.

[PIN 00a] PINHEIRO, M.K.; LIMA, J.V.; BORGES, M.R. S. **A Framework for Awareness in CSCW Systems**. Disponível em: <http://www.inf.ufrgs.br/~manuele/ArtigoAwareness-25pag.doc.zip>. Acesso em: mar.2000.

[PIN 00b] PINHEIRO, M.K.; LIMA, J.V.; BORGES, M.R. S. **Awareness in Groupware Systems**. Disponível em: <http://www.inf.ufrgs.br/~manuele/awareness.zip>. Acesso em: out.2000.

[PRA 99] PRAKASH, Atul; SHIM, Hyongsop; LEE, Jang Ho. Data Management issues and tradeoff in cscw systems. **IEEE Transactions on Knowledge and Data Engineering**, New York, v.11, n.1, p. 19-51, Jan./Fev. 1999.

[PRE 95] PREECE, J. **A guide to usability:** human factors in computing**.** Addison-Wesley/The Open University, 1995.

[PRI 99] PRINZ, Wolfgang. **The POLITeam Awareness Client**. Disponível em: <http://orgwis.gmd.de/projects/POLITeam/poliawac/>. Acesso em: out.1999.

[PRO 99] PROJECT POLITEAM. **POLITeam**. Disponível em: <http://orgwis.gmd.de/projects/POLITeam/>. Acesso em: out.1999.

[QUI 99a] QUINT, Vincent;VATTON, Irène**. Thot Tool Kit API**. Disponível em: <http://www.inrialpes.fr/Thot/Doc/APIman.html>. Acesso em: fev.1999.

[QUI 99b] QUINT, V.; VATTON, I. **Introduction to Amaya**. Disponível em: <http://www.w3c.org/pub/WWW/TR/NOTE-Amaya-970220>. Acesso em: fev.1999.

[RAM 98] RAMDUNY, Devina; DIX, Alan; RODDEN, Tom. Exploring the design space for notification servers. In: CSCW, 1998, Seattle, Washington. **Proceedings...**. Seattle: ACM Press, 1998. p. 227-235.

[RAP 99] RAPOSO, Alberto B.; MAGALHÃES, Léo P.; RICARTE, Ivan L. M. Interação na Web. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA (JAI) - CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC), 19.,1999, v.II. Rio de Janeiro, RJ. **Anais...** Rio de Janeiro: Edição EntreLugar, 1999. p. 1-50.

[RED 00] REDDY, Surendra; FISHER, Mark Leighton. **Requirements for Event Notification Protocol.** Internet Draft. Disponível em: <http://search.ietf.org/proceedings/99mar/I-D/ draft-ietf-webdav-enpreq-01.txt>. Acesso em: jan.2000

[RHY 92] RHYNE, James R.; WOLF, Catherine G. Tools for supporting the collaborative process. In: ANNUAL SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY – UIST, 5., 1992. Montrey, California, **Proceedings…** Montrey:ACM Press, 1992. p. 161-170.

[ROD 96] RODDEN, Tom; PALFREYMAN, Kevin. A protocol for user awareness on the world wid web. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK – CSCW, 1996, Massachusetts, **Proceedings…** Disponível em: <http://www.comp.lancs.ac.uk/ computing/users/kev/computing/project>. Acesso em: set.1999.

[ROS 96] ROSEMAN, Mark; GREENBERG, Saul. Building real time groupware with GroupKit, a groupware toolkit. **ACM Transactions on Computer Human Interactions**, New York, v.1, n.3, p. 66-106, Mar. 1996. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: jan.2000.

[ROS 97] ROSEMAN, Mark; GREENBERG, Saul. Building groupware with GroupKit. In: HARRISON, M. (Ed.). **Tcl/Tk Tool.** O'Reilly Press, 1997. p.535-564. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: jan.2000.

[RUM 94] RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W. *et al*. **Modelagem e Projetos Baseados em Objetos**. Rio de Janeiro: Campus, 1994.

[RUS 98] RUSSEL, Susan; KRAUT, Robert E.; LERCH, Javier; SCHERLIS, Willian L.; MCNALLY, Matthewm; CADIZ, Jonathan J. Coordination, overload and team performance: effects of team communication strategies. In: CSCW, 1998, Seattle, Washington. **Proceedings...**. Seattle: ACM Press, 1998.p. 275-288

[SAL 98] SALCEDO, Manuel Romero**. Alliance sur l'Internet**: support pour l' édition coopérative des documents struturés sur un réseau à grande distance**.** Grenouble, France: INPG, 1998.

[SCH 98] SCHMIDT, Albercht; SPECKER, Alexander; PARTSH, Gerhard; WEBER, Michael; HÖCK, Siegfield. An Agent-Based Telecooperation Framework. COBUILD, 1998. **Proceedings...** Darmstadt, 1998. Disponível em: <http://www-vs.informatik.uni-ulm.de/Papers/cobuild98/cobuild98.ps>. Acesso em: out.1998.

[SIE 94] SIEMIENIUCH, C.E.; SINCLAIR, M.A. Concurrent Engineering and CSCW: the human factor. In SPURR, Kathy; LAYZELL, Paul; JENNISON, Leslie; RICHARDS, Neil. **Computer Support for Co-Operative Work.** Chichester, EUA: John Wiley, 1994.

[SIL 89] SILBERSCHATZ, Abraham; KORTH, Henry F. **Sistema de Bancos de Dados**. Sao Paulo: Mcgraw-Hill, 1989.

[SMG 96] SMITH, Ian. Toolkits for multimedia awareness. In: CHI, 1996. **Proceedings**… Disponível em: <http://www.acm.org/sigchi/chi96/ proceedings/doctoral/Smith/ies_txt.html>. Acesso em: jul.1999.

[SMI 98] SMITH, Gareth; O' BRIEN, Jon. Re-coupling tailored user interfaces. In: CSCW, 1998, Seattle, Washington. **Proceedings...**. Seattle:ACM Press, 1998. p. 237-246.

[SOH 98] SOHLENKAMP, Markus**. Supporting group awareness in multi-user enviroment through perceptualization**. Paderborn: Fachbereich Mathematik-Informatik der Universität - Gesamthochschule, 1998. Disponível em: <http://orgwis.gmd.de/projects/POLITeam/poliawac/ ms-diss/>. Acesso em: jul.1999.

[SOU 96] SOUZA, Adriana Silveira de. **Um Estudo Sobre Trabalho Cooperativo Suportado Por Computador (CSCW):** Trabalho individual. Porto Alegre: CPGCC/UFRGS, 1996.

[SPU 94] SPURR, Kathy; LAYZELL, Paul; JENNISON, Leslie; RICHARDS, Neil. **Computer Support for Co-Operative Work.** Chichester, EUA: John Wiley, 1994.

[TAM 97] TAMMARO, S. G.; MOISIER, J. N.; GOODWIN, N. C.; SPITZ, G. Collaborative Writing is hard to support: a field study of collaborative writing. **Computer Supported Cooperative Work: The Journal of Collaborative Computing,** v.6, n.1, p. 19-51, 1997.

[VEE 97] VEERTEGALL, Roel; VELICHKOVSKY, Boris; VAN DER VEER, Gerril. Catching the Eye. **ACM SIGCHI Bulletin,** New York, v.29, n.4, out., 1997. Disponível em: <http://www.acm.org/sigchi/bulletin/1997.4/vertegal.html>. Acesso em: set.1999.

[VEN 98] VENNERS, B. Introduction to design techniques. **JavaWorld,** fev., 1998. Disponível em: <http://www.javaworld.com/jw-02-1998/jw-02-techniques.html>. Acesso em: mai.1999.

[VES 98] VESSONI, M.; TREVELIN; C. L. Aspectos de Design de Ferramentas para Escrita colaborativa. In: CONFERÊNCIA LATINO-AMERICANA DE INFORMÁTICA, 24., 1998, Quito, Equador. **Memórias...** Quito, 1998.

[WAH 94] WAHL, T.; ROTHERMEL, K. Representing Time in Multimedia Systems. In: INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING AND SYSTEMS, 1994. **Proceedings...** Disponível em: <http://www.informatik.uni-stuttgard.de/ipvr/vs/Publications/Publications.html>. Acesso em: jun.1999.

[WEB 99] WEBDAV: WWW Distributed Authoring and Versioning. In: INTERNET ENGINEERING TASK FORCE MEETING, 44., 1999, Minneapolis, Minesota. **Proceedings...** Disponível em: <http://search.ietf.org/proceedings/99mar/44th-99mar-ietf-36.html>. Acesso em: jan.2000.

[WIL 94] WILLIANS, Neil; BLAIR, Gordan S.; COULSON, Geoff; DAVIES, Nigel; RODDEN, Tom. The Impact of Distributed Multimedia systems on CSCW. In: SPURR, Kathy; LAYZELL, Paul; JENNISON, Leslie; RICHARDS, Neil. **Computer Support for Co-Operative Work.** Chichester, EUA: John Wiley, 1994.