ARIANE: UM MECANISMO DE APOIO À PERCEPÇÃO
EM BASES DE DADOS COMPARTILHADAS

Vaninha Vieira dos Santos

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por:

Profª. Cláudia Maria Lima Werner, D.Sc.

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Prof. Marcos Roberto da Silva Borges, Ph.D.

Profª. Karin Becker, Docteur

RIO DE JANEIRO, RJ - BRASIL

OUTUBRO DE 2003

SANTOS, VANINHA VIEIRA DOS

Ariane: Um Mecanismo de Apoio à Percepção em Bases de Dados Compartilhadas [Rio de Janeiro] 2003

XI, 103 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Percepção

2. Trabalho Cooperativo Apoiado por Computador

3. Banco de Dados

I. COPPE/UFRJ II. Título (série)

"...em cada época podemos viver
uma vida interessante e útil.
O indispensável é não
desperdiçarmos a vida e podermos
dizer: "Fiz o que pude".
Aqui está tudo quanto o mundo
tem o direito de exigir de nós
e a única coisa que nos dará
um pouco de felicidade."

- Madame Curie -
(Marya Sklodowska Curie)

# AGRADECIMENTOS

Desde que decidi ingressar em um mestrado, começando aí a saga por diversos programas, em diferentes lugares, percebi que sozinha eu não iria conseguir chegar muito longe. E, graças a Deus, consegui muita ajuda, de diversas pessoas. Isso não foi diferente no decorrer do curso, durante as disciplinas, ao final delas, até a definição de uma área de estudo, chegando a uma proposta de tese, reduzindo o escopo para chegar a um trabalho "factível", vivenciando "dramas existenciais", até chegar ao momento atual: a defesa. Muita gente participou desse processo ativamente comigo. Foram pessoas altamente presentes e fundamentais para que essa carta pudesse estar sendo escrita. Citar todos é uma tarefa difícil, foram tantos, espero não cometer injustiças. Além disso, o mestrado para mim não significou apenas a realização do curso na COPPE, mas todo o processo de sair de casa, vir para uma outra cidade, viver com pessoas diferentes, que pensam e agem de forma bastante diferente. Essa experiência contribuiu muito para a minha formação e realização pessoal. Às pessoas que participaram comigo desse processo, nesses quase 3 anos, também devo o meu agradecimento.

Dessa maneira, eu agradeço:

A **Deus**, força maior que rege a nós e ao universo, que me deu o principal para estar aqui: vida e saúde.

A **minha mãe Florisa** e a **meu pai José Milton** (*in memorian*), que foram a melhor escola que pude desejar. Embora não tenham tido muito acesso a estudo, eles se mostraram muito letrados na escola da vida e são os grandes responsáveis pelos principais traços que me tornaram o que sou hoje.

A **Kilza**. Sua presença constante certamente tornou muito mais fácil e agradável vivenciar esse período longe de casa, da família e dos amigos. Sua força e incentivo foram decisivos para me sustentar nos momentos de angústia e desânimo.

Agradeço do fundo do coração às minhas queridas orientadoras, **Profª Marta Mattoso** e **Profª Cláudia Werner**, sempre muito carinhosas comigo, pelos conhecimentos transmitidos, pelas palavras de incentivo, pelas idéias compartilhadas. Ouvi de muitas pessoas que fizeram mestrado comentários de que os dias de reunião com os orientadores eram dias de *stress* absoluto. Eu posso dizer que esses eram os dias em que eu me sentia mais empolgada e motivada a seguir adiante. Em geral, eu sempre chegava às reuniões em crises existenciais e saía com a sensação de que tudo terminaria bem e de que eu estava em excelentes mãos.

A **Mangan**, meu co-co-orientador, que foi uma peça fundamental neste trabalho e a seu parceiro constante: o caderninho! Valeu Mangan! Pelas idéias que deram origem a esta dissertação, pelos diversos comentários sempre pertinentes, pelas broncas, pelos ensinamentos, pelas revisões, e por tentar manter sempre o espírito de grupo e de colaboração, embora saibamos que passar da teoria para a prática nesse quesito não é fácil. Para não perder o hábito: Avancemos! ;-)

Ao **Prof. Marcos Borges** e à **Profª Karin Becker**, por aceitarem fazer parte dessa banca, mesmo com tantos compromissos.

À **Profª Maria Luiza Campos**, por sua atenção, carinho, e pelas dicas valorosas sobre OLAP e sistemas de apoio à decisão.

Toda essa história começou muito antes do mestrado em si, já na fase de seleção. Neste processo, tenho muito, mas muito mesmo a agradecer a três pessoas: a **Claudete**, à **Profª Christina** e à **Profª Aline**. Claudete, você é uma das pessoas que eu mais admiro, como profissional, mas principalmente como ser humano. Você é o que eu gostaria de ser quando crescer.

A **José Roberto Blaschek**, pelas oportunidades profissionais que me foram concedidas, pelas conversas, pelas dicas de vida, pelas palavras de incentivo e pela confiança. Tivemos pouca oportunidade de trabalhar diretamente, mas o pouco tempo já foi suficiente para aprender muito. Valeu mesmo!

Um agradecimento especialíssimo **à galera do LENS** pela convivência animada e pelo alto astral. A troca de idéias sobre teses e outros tópicos colaboraram para um maior amadurecimento e para aliviar o *stress* nos períodos mais conturbados. Em especial, gostaria de agradecer a **Murta**, **Ana Paula**, **Aline**, **Gustavo**, **Márcio**, **Lopes, Dantas** e **Sômulo** (valeu pelos cafezinhos!).

Ao **pessoal dos projetos** que participei através da COPPETEC. Em especial a **Renata Araújo** (trabalhar com você, logo no meu primeiro projeto aqui no Rio, foi um privilégio), **Luciana Netto**, **Jonice** (a admirável "garota-artigo") e **Fernanda Baião**.

Aos **clientes** da **COPPETEC** nesses projetos, em especial ao **Cmte Barros** e ao **Cmte Arnaldo** (pelas idéias trocadas sobre as nossas teses, sobre OLAP, sobre o *Analysis Services*, pelo livro emprestado e pelas aporrinhações decorrentes das renovações desse empréstimo).

Aos **amigos da turma de calouros BD 2001** pelo convívio durante as disciplinas, ao término delas, pelos trabalhos conjuntos e pelas conversas e desabafos durante o período da tese. Em especial às meninas super poderosas **Dani** e **Marcinha**, aos "baianos" **Pablito** e **Manolo** (ôxe!) e ao "mineirim" **Matheus**.

Ao pessoal que participou do projeto GOA-JDO e aos demais colegas da linha de Banco de Dados, em especial a **Robson Pinheiro**, **Gabriela Ruberg**, **Paulo Pires** e **Yoko** pela ajuda, idéias e incentivos ao trabalho.

A **José Maria** e a **Manuele Pinheiro** pela troca de idéias sobre percepção.

A todo o **corpo administrativo da COPPE**, em especial a **Patrícia Leal** que quebrou altos galhos e estava sempre disposta a ajudar no possível.

À conterrânea **Lourdes**, pelo cafezinho e pelo bate-papo na copa.

Ao **pessoal dos fóruns de discussão** na internet sobre JDO e AspectJ, em especial a **Patrice Thiebaud**, pelas idéias, sugestões, esclarecimento de dúvidas, etc. Essa forma de cooperação se mostra cada vez mais importante para o desenvolvimento de trabalhos acadêmicos e sem as sugestões recebidas certamente eu teria demorado um pouco mais para encontrar soluções para os problemas que fui encontrando no desenvolvimento do protótipo.

Ao **CNPq**, pelo apoio financeiro.

À **UFBA**, pela minha liberação.

A **Keila** e a **Laura** pela convivência, viagens, amizade e carinho.

A todos os membros queridos da **minha** extensa **família**, em especial à minha irmã **Delcina** e ao meu cunhado **Antonio José**, que se empolgaram com a realização do meu mestrado, deram força, incentivaram, e sempre perguntavam: quando é a defesa? Demorou, mas chegou.

Aos amigos conquistados na cidade maravilhosa, em especial a **Benny** e **Calixto**, pela acolhida assim que cheguei aqui, e por guiar essa baiana perdida pelas ruas do Rio para as entrevistas do mestrado. Também a **Lamis** e a **MOC** pelas aventuras, conversas e por agüentarem meu *stress* falando o tempo todo de Ariane.

À galera do 302 da Mascarenhas de Morais, **Melise**, **Isabel** e **Gil**.

À **cidade do Rio de Janeiro** e ao **bairro de Copacabana**, com seu calçadão, ruas movimentadas, noites sem fim, pessoas (muitas pessoas) interessantes e diferentes. Agradeço pela acolhida e declaro o meu amor eterno.

A todos aqueles que direta, ou indiretamente, contribuíram para a realização desse trabalho, vibraram, torceram, me agüentaram e me apoiaram.

Essa lista de agradecimento não tem grau de importância. Todas as contribuições, por menores ou insignificantes que pareçam a princípio, certamente têm uma finalidade e uma razão de ser. Todos vocês, inclusive os que por esquecimento não tenham sido citados, são muito importantes para mim e foram fundamentais para que mais essa etapa da minha vida tenha chegado a um final. Assim, deixo aqui eternizado meu muito, muitíssimo obrigado.

Beijinhos, já com saudades.

**Vaninha Vieira.**

# O FIO DE ARIANE

Segundo a mitologia grega, um jovem herói ateniense, chamado Teseu, ao saber que sua cidade deveria pagar a Creta um tributo anual composto de sete rapazes e sete moças, para serem entregues ao insaciável Minotauro que se alimentava de carne humana, solicitou ser incluído dentre eles. O Minotauro vivia em um labirinto, constituído de salas e passagens intrincadas do palácio de Knossos, cuja construção é atribuída ao arquiteto ateniense Dédalo. Ao chegar em Creta, Teseu conheceu Ariane, a filha do rei Minos, que se apaixonou por ele e, resolvida a salvar Teseu, pediu a Dédalo a planta do palácio. Ela acreditava que Teseu poderia matar o Minotauro, mas não saberia sair do labirinto. Ariane deu um novelo a Teseu recomendando que o desenrolasse à medida que entrasse no labirinto para encontrar a saída. Teseu usou essa estratégia, matou o Minotauro e, com a ajuda do fio de Ariane, encontrou o caminho de volta. Retornando a Atenas levou consigo a princesa. Depois de uma noite de amor, Teseu deixou-a na ilha de Naxos e ela nunca mais o viu.

Fonte: O Fio de Ariadne. In: http://teclec.psico.ufrgs.br/CD2/Nova/fioariadne.html

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# ARIANE: UM MECANISMO DE APOIO À PERCEPÇÃO EM BASES DE DADOS COMPARTILHADAS

Vaninha Vieira dos Santos

Outubro/2003

Orientadoras: Marta Lima de Queirós Mattoso
Cláudia Maria Lima Werner

Programa: Engenharia de Sistemas e Computação

Percepção em atividades colaborativas significa conhecimento e compreensão do que ocorre no contexto do grupo, que auxilia as pessoas a gerenciar suas atividades, ampliando a possibilidade de interação, gerando um trabalho mais coerente e coeso em menor tempo. Os mecanismos de percepção existentes, em geral, são implementados de forma bastante acoplada a aplicações cooperativas específicas, o que prejudica sua reutilização. Em ambientes colaborativos, a interação ocorre, geralmente, através da manipulação de artefatos compartilhados, os quais são comumente persistidos usando Sistemas de Gerência de Banco de Dados (SGBDs). Porém, os SGBDs não possuem mecanismos que apóiem a percepção sobre ações executadas nesses artefatos. Esta dissertação propõe um mecanismo, denominado Ariane, que visa apoiar a percepção a partir do monitoramento de SGBDs. Ariane se diferencia de outras abordagens por ser independente de aplicação ou SGBD específicos. Para isso, faz uso de tecnologias que provêem persistência transparente e da programação orientada a aspectos, que permite que o mecanismo seja acoplado a qualquer aplicação de forma não intrusiva. Além disso, Ariane propõe o uso de uma estrutura multidimensional, o cubo de percepção, para armazenamento das informações de percepção produzidas, de modo que consultas analíticas, que recuperem valor agregado dessas informações, possam ser realizadas através de ferramentas de processamento analítico (OLAP).

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ARIANE: A MECHANISM FOR AWARENESS SUPPORT
IN SHARED DATABASES

Vaninha Vieira dos Santos
October/2003

Advisors: Marta Lima de Queirós Mattoso
Cláudia Maria Lima Werner

Department: Computer Science and Systems Engineering

Awareness in collaborative activities means the knowledge and understanding about what happens in the group context that helps people to coordinate their own activities and enlarge the possibility of interaction between participants, generating a more coherent and consistent work in less time. In general, existing awareness mechanisms are implemented in a tightly coupled way related to specific groupware systems, which can harness its reuse. In collaborative environments, people interaction generally occurs through shared artifacts manipulation, which are commonly persisted using Database Management Systems (DBMSs). However, DBMSs do not offer awareness support about actions executed on these artifacts. This dissertation proposes an awareness mechanism, named Ariane, which aims to provide awareness information to collaborative applications monitoring their DBMS. Ariane differentiates from other proposals because it is independent of a particular groupware or DBMS. This is achieved through the use of technologies that provide transparent persistence and aspect oriented programming (AOP) that allows the awareness mechanism to be coupled to any aspplication in a non-intrusive way. Besides, Ariane proposes the use of a multidimensional representation, the awareness cube, of the awareness information generated, so analytical queries, that retrieve aggregate value of these information, may be realized using On-Line Analytical Processing (OLAP) tools.

# ÍNDICE

# *Capítulo I*

## *Introdução*

## I.1. MOTIVAÇÃO E CARACTERIZAÇÃO DO PROBLEMA

Mudanças nas necessidades de negócios, com tarefas e projetos cada vez mais complexos e prazos de execução cada vez menores, vêm demandando que a forma de trabalho das organizações seja alterada, com a substituição do esforço totalmente individual pela utilização de equipes, interagindo cooperativamente.

Devido a exigências do mercado, como a globalização, e à estruturação das organizações, é cada vez mais comum que as equipes sejam formadas por pessoas dispersas geograficamente, ou que, mesmo fisicamente próximas, tenham uma flexibilidade de horários, trabalhando em turnos distintos. Para que a realização do trabalho em equipe funcione de maneira eficaz, são necessárias forte cooperação e interação entre os indivíduos envolvidos. A comunicação e o entendimento dos objetivos e papéis de cada um dentro do grupo, bem como o acompanhamento do que está acontecendo no âmbito das atividades do grupo são requisitos fundamentais para garantir a qualidade do trabalho desenvolvido.

A evolução das tecnologias de computadores e de comunicação como a Internet, a comunicação sem fio, a comunicação via cabo, entre outras, têm viabilizado a ligação social entre as pessoas (ARAÚJO, 2000), oferecendo novas possibilidades de interação e atuando, dessa forma, como facilitador do trabalho em equipe (ELLIS et al., 1991).

A área de Trabalho Cooperativo Apoiado por Computador (CSCW – *Computer Supported Cooperative Work*) estuda o modo como as pessoas trabalham em equipe e procura descobrir como os computadores podem auxiliá-las na realização de suas tarefas (ELLIS et al., 1991).

A partir das pesquisas sobre CSCW, novas ferramentas e aplicações foram desenvolvidas para apoiar a cooperação. Estas aplicações, também chamadas de aplicações cooperativas ou sistemas de *groupware*, representam "a tecnologia

computacional que auxilia grupos na realização de suas tarefas, em diversos contextos de colaboração e comprometimento, oferecendo níveis distintos de **comunicação**, **coordenação**, **percepção** e manutenção da **memória do grupo**, de acordo com as necessidades e objetivos de interação de cada grupo" (ARAÚJO, 2000).

A **comunicação** refere-se a tecnologias que viabilizem a conectividade entre os participantes, a **coordenação** trata do acompanhamento do processo de colaboração, a **memória do grupo** é onde estão registradas as interações entre os membros do grupo e a **percepção** é o conhecimento e compreensão por parte dos participantes sobre o que ocorre no contexto do grupo, relativo tanto às pessoas quanto às atividades desenvolvidas. O foco desta dissertação está no conceito de percepção.

A habilidade de percepção do que ocorre com os outros membros de um grupo representa um aspecto importante para a naturalidade e fluidez da colaboração, auxiliando a reduzir as características de "estranheza" entre as pessoas, comum em comunicações remotas. A ausência dessa percepção pode gerar diversos problemas dentro do grupo, como a falta de motivação, conflitos e o trabalho duplicado ou inconsistente (PINHEIRO, 2001) (SOHLENKAMP, 1998).

A interação entre participantes de uma equipe, quando trabalham cooperativamente, ocorre, geralmente, através da manipulação de artefatos compartilhados. Para permitir a durabilidade e compartilhamento desses artefatos, a maioria das aplicações cooperativas utiliza a persistência em Sistemas de Gerência de Bancos de Dados (SGBDs).

Os SGBDs mantêm as informações sobre os artefatos compartilhados e sobre o ciclo de vida desses artefatos (o que mudou, quem executou a alteração, em que momento). O monitoramento das mudanças ocorridas sobre artefatos persistentes pode apoiar a percepção sobre as interações realizadas na execução das atividades do grupo. No entanto, os SGBDs, por serem ferramentas de propósito geral, não implementam os serviços necessários para apoiar a atividade colaborativa (BORGES et al., 2000), e isso inclui o suporte à percepção.

Manter a percepção sobre as atividades dos outros, embora pareça trivial e garantido no mundo real, mostra-se bastante difícil em aplicações cooperativas, onde os recursos de informação são pobres e os mecanismos de interação são desconhecidos. Esse problema é especialmente evidente quando se trata de percepção sobre interações

assíncronas, na qual os participantes não interagem em um mesmo instante de tempo, sendo informados sobre a interação em um momento posterior à sua realização. As aplicações cooperativas, em geral, falham no suporte a esse tipo de percepção (PINHEIRO, 2001)(BORGES e PINO, 1999)(SOHLENKAMP, 1998).

A maioria dos mecanismos de percepção existentes envolve soluções locais para problemas de domínio específico e isolam abordagens e princípios que são difíceis de generalizar em outras situações, obrigando os projetistas das aplicações cooperativas a recriar os mecanismos a cada nova aplicação (GUTWIN e GREENBERG, 2002).

Poucas são as soluções de percepção que apóiam diferentes aplicações. Dentre as existentes, percebe-se que, ou são desenvolvidas para funcionar sobre um SGBD específico, ou exigem que muitas mudanças sejam feitas na aplicação cooperativa para viabilizar sua utilização.

Uma outra questão a ser observada é que o volume de informações de percepção geradas ao longo das diversas interações tende a ser muito grande, ultrapassando a capacidade humana e a habilidade técnica para sua interpretação. Muito conhecimento sobre os participantes e as interações do grupo pode ser extraído do histórico de informações de percepção produzidas, utilizando ferramentas que apóiem essa análise. Entretanto, os mecanismos de percepção existentes não tratam suas informações de modo a possibilitar esse tipo de análise.

Mecanismos de percepção de propósitos genéricos, ou seja, não dependentes de uma aplicação particular, viabilizam que informações de percepção provenientes de diferentes aplicações possam ser confrontadas e analisadas. Essa possibilidade é interessante, pois os usuários de um grupo costumam utilizar diferentes aplicações para realização do seu trabalho.

## I.2. OBJETIVOS DA DISSERTAÇÃO

O objetivo principal desta dissertação é o desenvolvimento de um mecanismo de apoio à percepção, que amplie a oferta de informações de percepção em aplicações cooperativas, a partir do monitoramento do processo de persistência dessas aplicações. Uma premissa assumida é que as aplicações realizem persistência em SGBDs. Esse mecanismo, denominado **Ariane**, não deve gerar dependência de uma aplicação

específica, nem do SGBD utilizado pela aplicação, ou seja, Ariane deve poder ser reutilizado por diferentes sistemas em contextos distintos.

Ariane visa prover informações que apóiem, especialmente, a percepção em interações assíncronas. Isso se dá porque os tipos de operações tratados por Ariane referem-se a ações relacionadas a artefatos persistentes das aplicações. Ações operacionais, como movimentação do mouse ou do cursor, ou ações sobre artefatos mantidos apenas em memória, fundamentais para apoiar interações síncronas, não são consideradas em Ariane. No entanto, em uma proporção menor, também as interações síncronas podem se beneficiar dos serviços de percepção de Ariane, uma vez que mudanças ocorridas na base de dados podem ser propagadas a usuários conectados ao espaço de trabalho tão logo ocorram.

Como nas interações assíncronas os usuários podem se ausentar por um longo período de tempo do espaço de trabalho, as informações de percepção devem ser persistidas em um repositório histórico. Sobre esse histórico de interações deve ser possível realizar consultas analíticas que extraiam valor agregado das informações de percepção produzidas.

Dessa maneira, dentre as funcionalidades principais de Ariane temos: (i) identificar e representar ações relevantes sobre SGBDs executadas por usuários de uma aplicação cooperativa; (ii) monitorar de forma implícita, ou seja, sem que haja interferência no funcionamento habitual da aplicação ou do SGBD, as ações ocorridas sobre os artefatos persistentes da aplicação, gerando as informações de percepção; (iii) armazenar no repositório histórico as informações de percepção geradas; (iv) propagar as informações de percepção a componentes visuais de percepção registrados, para que a informação seja tratada e apresentada de acordo com as necessidades específicas dos usuários da aplicação; e (v) possibilitar que ferramentas de processamento analítico (OLAP – *On-Line Analytical Processing*) sejam utilizadas sobre o histórico de interações, para realizar análises e extrair conhecimento.

## I.3. CONTEXTO DA DISSERTAÇÃO

O desenvolvimento desta dissertação foi motivado pelo projeto *OdysseyShare* (WERNER et al., 2002), que tem como característica principal a definição e construção de um ambiente cooperativo para o desenvolvimento de *software* baseado em

componentes. O projeto *OdysseyShare* combina técnicas utilizadas no contexto do DBC (Desenvolvimento Baseado em Componentes), CSCW e Banco de Dados e tem como objetivo explorar os aspectos colaborativos, por meio da construção de mecanismos que apóiem a interação de equipes em um ambiente de DBC.

As contribuições da área de banco de dados para esse projeto incluem prover meios para armazenamento e recuperação de componentes, através do gerente de objetos GOA (SOUZA et al., 2002), a investigação de funcionalidades que poderiam ser acopladas ao SGBD para apoiar o desenvolvimento do ambiente colaborativo (VIEIRA et al., 2002) e a implementação do apoio à percepção, através do desenvolvimento de um protótipo de Ariane e da utilização desse protótipo junto ao ambiente *OdysseyShare*.

## I.4. ORGANIZAÇÃO DA DISSERTAÇÃO

O restante desta dissertação está dividida nos seguintes capítulos:

O **Capítulo 2** apresenta algumas definições sobre CSCW e aplicações cooperativas, faz uma revisão sobre o conceito de percepção nesse tipo de aplicação e discute alguns trabalhos relacionados a esta dissertação.

No **Capítulo 3** é apresentada a proposta do mecanismo de percepção Ariane, o processo utilizado para apoiar a percepção em aplicações cooperativas, a arquitetura definida para o mecanismo de percepção e as suas principais funcionalidades.

O **Capítulo 4** descreve a implementação de um protótipo de Ariane, discute algumas tecnologias que foram empregadas para apoiar esse desenvolvimento, especialmente para garantir o requisito de independência de aplicação cooperativa, SGBD e modelo de dados, detalha como as informações de percepção produzidas por Ariane foram modeladas para possibilitar sua utilização em sistemas de processamento analítico (OLAP), e apresenta um exemplo de utilização de Ariane junto ao ambiente colaborativo *OdysseyShare*.

O **Capítulo 5** encerra a dissertação discutindo as principais contribuições de Ariane, as limitações da proposta e do protótipo e as melhorias que poderão ser realizadas, em trabalhos futuros, sobre o protótipo e para avançar nas pesquisas sobre o tema.

# *Capítulo II*

## *CSCW e Percepção*

*Este capítulo apresenta alguns conceitos sobre CSCW, suas principais características, sua relação com SGBDs e detalha o conceito de percepção. São discutidas as principais características da percepção em ambientes de colaboração e algumas questões que devem ser levadas em consideração quando da construção de um mecanismo de apoio à percepção. Alguns mecanismos existentes na literatura são apresentados e comparados, analisando-se pontos onde os mesmos falham. O mecanismo de percepção proposto nesta dissertação faz parte da categoria de mecanismos de percepção de propósito genérico. Ele visa apoiar a percepção do espaço de trabalho compartilhado e está voltado, especialmente, para interações assíncronas e para apoiar usuários com papel de coordenador, operador e analista. As informações de percepção são estruturadas segundo a representação 5W+1H. Questões específicas de apresentação da informação de percepção e soluções para problemas como sobrecarga e privacidade não são tratadas nesta dissertação.*

### II.1. CARACTERÍSTICAS DE CSCW

A área de Trabalho Cooperativo Apoiado por Computador (CSCW – *Computer Supported Cooperative Work*) estuda como as pessoas interagem para a realização de trabalho cooperativo e como a tecnologia dos computadores pode auxiliá-los nessa interação (ELLIS et al., 1991). Encontrar uma definição precisa para o que é trabalho cooperativo não é fácil. Em (RAMAMPIARO e NYGARD, 1999), é citada uma definição apresentada por Karl Marx que diz que "quando vários trabalhadores trabalham em conjunto, lado a lado, de acordo com um plano, em um mesmo processo, ou diferentes, mas conectados, essa forma de trabalho é chamada cooperação". Essa definição, entretanto, não está completa, além de assumir que o trabalho cooperativo é sempre planejado. Assim, uma definição mais genérica e intuitiva pode ser: "um processo de trabalho envolvendo diversas pessoas agindo em conjunto, em um contexto

compartilhado, para realizar tarefas com objetivo comum" (RAMAMPIARO e NYGARD, 1999).

CSCW é uma área multidisciplinar com origens nas ciências sociais, envolvendo disciplinas como antropologia, sociologia e psicologia, e nas tecnologias de computadores, envolvendo profissionais de áreas como engenharia de software, banco de dados, interação homem-máquina, inteligência artificial e redes de computadores.

As pesquisas desenvolvidas em CSCW deram origem a uma nova categoria de aplicações, chamadas aplicações cooperativas ou sistemas de *groupware*. Essas aplicações objetivam aumentar o potencial dos grupos, fazendo com que o produto resultante das interações apresente melhor qualidade do que a soma das contribuições individuais de cada membro do grupo (ARAÚJO, 2000). Além disso, elas auxiliam a manutenção do conhecimento corporativo através de uma infraestrutura, conhecida como memória organizacional, ou memória do grupo, capaz de manter o conhecimento sobre o negócio da organização, através do armazenamento das interações entre os participantes (ARAÚJO, 2000).

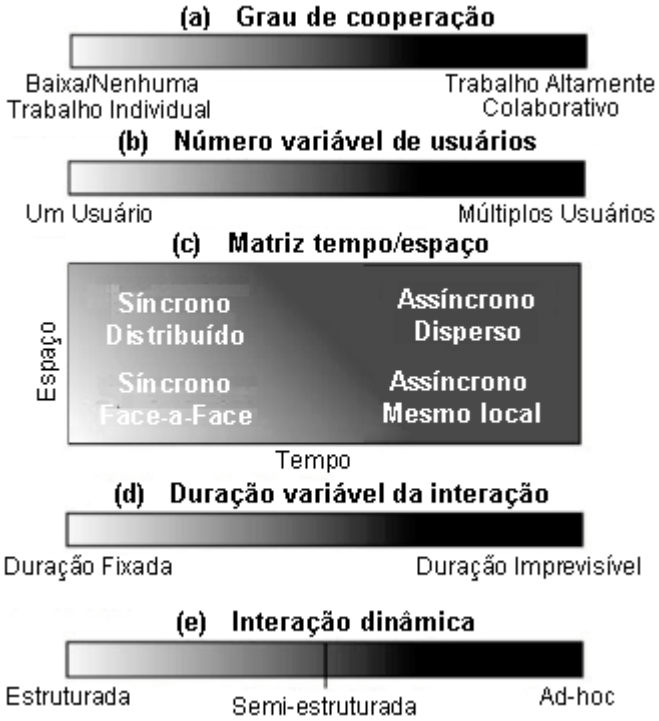



**Figura 1 - Características do trabalho cooperativo (RAMAMPIARO e NYGARD, 1999)**

Como mostra a Figura 1, os projetistas de aplicações cooperativas devem considerar as seguintes características do trabalho cooperativo (RAMAMPIARO e NYGARD, 1999):

(a) **Grau de cooperação**: pode variar de acordo com a necessidade da situação, sendo necessária a cooperação total em alguns casos, ou nenhuma em outros;

(b) **Número variável de usuários**: o número de pessoas trabalhando não é fixo, podendo haver uma pessoa em determinado momento e várias em outro;

(c) **Matriz tempo/espaço**: as pessoas podem trabalhar no mesmo horário, ou em horários diferentes, além disso, podem estar em um mesmo espaço físico ou dispersas geograficamente;

(d) **Duração variável da interação**: nem sempre é possível estimar o tempo necessário de interação previamente, sendo essa duração bastante variável;

(e) **Interação dinâmica**: as atividades cooperativas podem, em um curto espaço de tempo, trabalhar com dados estruturados, semi-estruturados ou *ad-hoc*.

Dentre essas características, a matriz tempo/espaço (Figura 1 – item c) é utilizada para classificar a natureza das aplicações cooperativas (ELLIS et al., 1991) do ponto de vista de sua capacidade em quebrar as fronteiras de tempo e localização entre indivíduos e estabelecer a comunicação entre eles. Quanto ao espaço, os indivíduos podem interagir estando no mesmo local (interação localizada) ou geograficamente dispersos (interação distribuída). Quanto ao tempo, a interação entre os participantes pode ocorrer basicamente de três formas:

- **Síncrona**: os participantes trabalham durante o mesmo período de tempo e têm conhecimento imediato do trabalho produzido pelos outros. Exemplo dessa interação é o trabalho realizado durante uma reunião. Ferramentas de reunião como o NetMeeting (MICROSOFT, 2003b) e de comunicação instantânea, como o ICQ (ICQ, 2003) apóiam esse tipo de interação;
- **Assíncrona**: os participantes não trabalham no mesmo período de tempo e, conseqüentemente, não tomam conhecimento sobre o trabalho produzido pelos outros participantes em tempo real. Modificações sobre artefatos compartilhados são observadas pelos membros imediatamente, se os mesmos estiverem conectados, ou pode haver um intervalo até que os mesmos se reconectem ao

sistema. Esse tipo de interação ocorre, por exemplo, no desenvolvimento cooperativo de software. Ferramentas de gerência de grupos de discussão como o YahooGroups (YAHOO, 2003) e de espaço de trabalho compartilhado como o BSCW (GMD, 2002), apóiam interações assíncronas;

- **Multi-síncrona**: nesse tipo de interação, cada membro tem uma cópia do dado compartilhado, modificando suas cópias em paralelo, de modo a atingir seus objetivos mais rapidamente (MOLLI et al., 2002). A junção dos trabalhos paralelos implica em tratar problemas de consistência entre as várias cópias do artefato compartilhado. Esse tipo de interação também ocorre, freqüentemente, no desenvolvimento cooperativo de software. Exemplos de sistemas que apóiam esse tipo de interação são as ferramentas de gerência de configuração, como o CVS (CVS, 2003).

Aplicações cooperativas como o SAMS (MOLLI et al., 2002) e o *OdysseyShare* (WERNER et al., 2003), propõem suporte para essas três formas de interação.

A atividade colaborativa envolve, basicamente, quatro aspectos principais, inter-relacionados que devem ser oferecidos pelas aplicações cooperativas (Figura 2): a **comunicação** entre os membros do grupo, a **coordenação** de suas atividades, o armazenamento e recuperação do conhecimento comum através da **memória do grupo** e a **percepção** do contexto do trabalho do grupo, por seus membros (BORGES e PINO, 2000) (ARAÚJO, 2000).

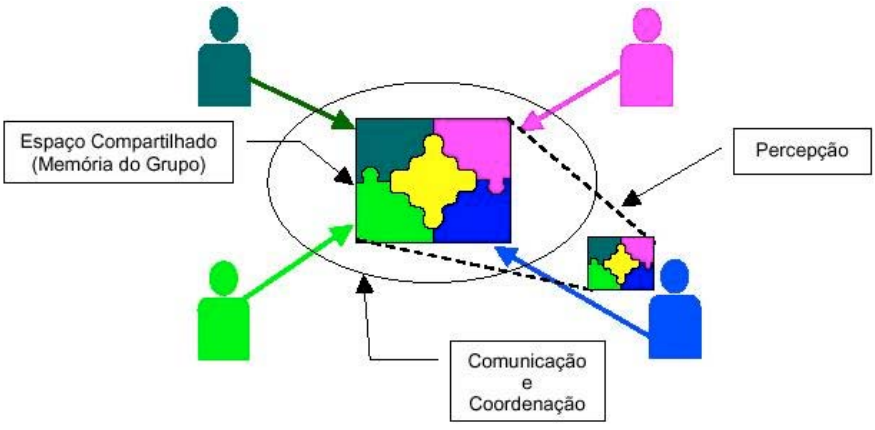



**Figura 2 - Esquema geral dos aspectos do trabalho em grupo (ARAÚJO, 2000)**

O primeiro obstáculo ao compartilhamento é vencer o isolamento e a distância entre os membros da equipe de trabalho, estabelecendo a **comunicação** entre as partes

envolvidas. Essa comunicação pode se dar de duas maneiras (Figura 3): direta, ou indiretamente. A comunicação direta ocorre por iniciativa dos próprios participantes, os quais contatam-se uns aos outros utilizando dispositivos como telefone ou correio eletrônico. A comunicação indireta ocorre, especialmente, nas interações assíncronas e se caracteriza pela manipulação do histórico de ações de artefatos compartilhados, os quais estão disponíveis em um espaço de trabalho comum. Esse histórico permite conhecer o que houve e o que está acontecendo ao artefato, e, conseqüentemente, dentro do grupo. Em (DOURISH, 1997), o artefato compartilhado é visto, essencialmente, como o único espaço compartilhado disponível aos participantes, representando a informação chave na colaboração assíncrona.


**Figura 3 – Formas de comunicação entre membros de um grupo**

A **coordenação** refere-se ao acompanhamento da execução do processo de cooperação, especificando como se dará a interação, definindo regras e limites, estipulando responsabilidades e controlando a execução das tarefas, visando garantir a produtividade e o sucesso dos objetivos do grupo.

A **memória do grupo**, ou memória organizacional, é o registro persistente de todas as interações ocorridas no grupo.

Por fim, a **percepção** é a consciência das ações executadas pelos demais participantes. Devido a sua importância no contexto deste trabalho, esse conceito será detalhado, ao longo deste capítulo.

## II.2. GERENCIAMENTO DE DADOS COMPARTILHADOS (CSCW X SGBD)

Membros de um grupo, quando trabalham cooperativamente, necessitam trocar conhecimentos. Um requisito básico para auxiliar a atividade colaborativa é prover

mecanismos de interação que tornem essa troca, inclusive, persistente. Sem a persistência, a interação é efêmera e não pode ser compartilhada com pessoas que não estejam envolvidas no momento em que a mesma ocorreu (BORGES e PINO, 2000).

Do ponto de vista da necessidade de persistência, os mecanismos que apóiam a comunicação entre os participantes de um grupo podem ser classificados em, pelo menos, três categorias (BORGES et al., 2000):

- **Interfaces síncronas**: desenvolvidas para apoiar troca de mensagens instantâneas e efêmeras, necessitam baixa ou nenhuma persistência. A persistência, nesse caso, ocorre somente na memória humana, ou através da gravação da interação para posterior exibição a usuários que não tenham participado da mesma;
- **Mensagens de correio eletrônico**: os dados tratados podem ser permanentes, porém são imutáveis. Além disso, sistemas de correio eletrônico, geralmente, suportam apenas informações não estruturadas e provêm mecanismos de recuperação limitados;
- **Memória compartilhada**: abrangem dados perenes que, na maioria dos casos, devem permanecer ativos ao longo de todo o projeto. A memória compartilhada, em geral, é mantida por SGBDs.

Por serem ferramentas de propósito geral, os SGBDs falham em prover funcionalidades apropriadas para lidar com interações entre membros de um grupo para alcançar um conhecimento comum (MARIANI e RODDEN, 1991) (MARIANI, 1997) (RAMAMPIARO e NYGARD, 1999) (PRAKASH et al., 1999) (PREGUIÇA et al., 2000) (BORGES et al., 2000). Isso se deve, em grande parte, à incompatibilidade entre os princípios que regem os SGBDs e algumas demandas das aplicações cooperativas, tais como:

- Conflitos em aplicações cooperativas são vistos como possibilidades de interação, pois indicam interesses de usuários diferentes em artefatos comuns. Dessa maneira, nas aplicações cooperativas os usuários são encorajados a trabalhar simultaneamente sobre a mesma informação, o que praticamente é proibido pelos SGBDs convencionais;
- Os mecanismos de controle de consistência dos SGBDs, baseados em bloqueio dos dados e no modelo de transações convencionais, se preocupam em garantir

que os usuários estejam isolados das atividades uns dos outros, impedindo que os membros dos grupos percebam o que os outros estão fazendo;

- As atividades cooperativas tendem a ser de longa duração, podendo durar meses até que uma versão de um produto seja considerada satisfatória e aprovada. Dessa forma, o modelo de transações convencional, baseado em várias atividades de curta duração, não é adequado;
- O mecanismo de *log* dos SGBDs atuais se limitam a registrar as mudanças de estado ocorridas nos artefatos armazenados, para fins de recuperação de um estado consistente do banco em caso de falhas. Nenhuma notificação automática é feita para que os usuários desse banco tomem conhecimento dessas mudanças.

Com isso, esses sistemas são utilizados, meramente, como repositórios passivos de dados e a maioria das aplicações CSCW acaba por implementar mecanismos próprios para a criação e o gerenciamento das informações compartilhadas, o que demanda um grande esforço e retrabalho dos desenvolvedores dessas aplicações.

Como exemplo de problemas em utilizar SGBD para gerenciar informações em aplicações cooperativas, em (WEAR et al., 1995), são apresentados alguns desafios em projetar um banco de dados para um sistema de edição colaborativa de documentos:

- Implementação de um método de particionamento que quebre documentos em seções gerenciáveis;
- Manutenção dessa estrutura particionada à medida que mudanças sejam realizadas por cada usuário;
- Atualização da cópia mestra, quando necessário;
- Determinação se dois usuários estão tentando modificar a mesma seção.

Assim, o uso de SGBD para apoiar aplicações cooperativas impõe a revisão de diversos tópicos implementados nesses sistemas, tais como os mecanismos de controle de concorrência, problemas temporais, regras ativas e evolução de esquema (MARIANI e RODDEN, 1991) (PRAKASH et al., 1999) (BORGES e PINO, 2000) (PREGUIÇA et al., 2000).

Em (VIEIRA et al., 2002), são apresentados os principais requisitos gerados por aplicações cooperativas nos SGBDs, bem como sugestões de mudanças nos SGBDs

para apoiar esse tipo de aplicação. Dentre esses requisitos, o suporte à percepção é o tema tratado nesta dissertação e será discutido a partir da próxima seção.

## II.3. PERCEPÇÃO

A percepção é considerada uma das principais funcionalidades das aplicações cooperativas e um fator relevante para ampliar a usabilidade dessas aplicações (GUTWIN e GREENBERG, 2002). Esse tema tem recebido considerável atenção por parte dos pesquisadores das áreas de CSCW e Interação Homem-Computador (HCI - *Human Computer Interaction*). Sua importância é evidenciada por diversos trabalhos de pesquisa (DOURISH e BELLOTTI, 1992) (SOHLENKAMP, 1998) (BORGES e PINO, 1999) (ARAÚJO, 2000) (PINHEIRO, 2001) (GUTWIN e GREENBERG, 2002) e workshops, em importantes conferências, dedicados ao tema (CHI'97, CSCW'00, CHI'00).

Para melhor entender o termo percepção e sua importância no contexto do trabalho cooperativo, deve-se verificar como o mesmo é apresentado e definido na literatura. Uma das primeiras, e mais referenciadas, definições sobre percepção foi apresentada em (DOURISH e BELLOTTI, 1992), para os quais percepção é "uma compreensão das atividades dos outros, que provê um contexto para atividades próprias. Esse contexto é utilizado para garantir que as contribuições individuais sejam relevantes para as atividades do grupo como um todo, e para avaliar as ações individuais em relação aos objetivos e progresso do grupo". Atividades vão desde movimentos físicos da pessoa em uma sala (entrada e saída de uma sala ou prédio) até interações do participante com uma aplicação cooperativa (BORGES et al., 2000).

Em (SOHLENKAMP, 1998), percepção é definida como uma compreensão do estado de um sistema, incluindo atividades passadas, estado atual e opções futuras, e como um estado mental de um usuário (abrangendo a compreensão, o conhecimento, e o "estar atento"), o qual envolve a atividade de outros e provê um contexto para atividades próprias.

De uma forma geral, pode-se entender percepção como sendo o conhecimento e a compreensão de tudo o que ocorre dentro e fora de um sistema, relevantes para o desenvolvimento das atividades dos participantes do grupo, desempenhando seus diferentes papéis. Como sistema, entenda-se todo o contexto do grupo, incluindo os

artefatos compartilhados e aspectos sociais e organizacionais (que pessoas compõem o grupo, quais são suas atribuições e habilidades).

Os termos percepção e mecanismos de percepção são, muitas vezes, empregados de forma similar na literatura, como se fossem o mesmo conceito. Entretanto, deve-se fazer uma diferenciação entre os dois. O primeiro é um estado mental do usuário, enquanto o segundo corresponde às técnicas empregadas por um sistema para oferecer informações de percepção que auxiliem os membros do grupo a alcançar esse estado mental (SOHLENKAMP, 1998). Como exemplo dessa diferença consideremos a situação onde um novo usuário é introduzido em um grupo. A informação de percepção é "o usuário X ingressou no grupo". Os mecanismos de percepção são os meios promovidos pela aplicação para que os membros do grupo recebam essa informação. A percepção é alcançada quando os participantes do grupo recebem, processam e compreendem essa informação.

### II.3.1. Benefícios em Prover percepção

Segundo Gutwin e Greenberg (GUTWIN e GREENBERG, 2002), a percepção é vista como fator determinante para o sucesso da utilização de aplicações cooperativas. Manter os usuários atentos ao que ocorre dentro do grupo é fundamental para um bom andamento das atividades individuais e para a coordenação do trabalho como um todo.

A percepção é essencial para o fluxo e a naturalidade do trabalho, auxiliando a diminuir as sensações de trabalho impessoal e de distância, comuns nos ambientes virtuais (GEROSA et al., 2001). Através dessas informações, os participantes podem montar o contexto de suas atividades em relação às dos demais, coordenando-se para que os esforços de comunicação e de trabalho sejam revertidos em cooperação. Além disso, o coordenador do grupo pode valer-se dessas informações, que lhe fornecerá uma visão geral do andamento dos trabalhos e o estado de cada atividade, para motivar e organizar as pessoas dentro do grupo, encorajando a cooperação (GEROSA et al., 2001).

Quando trabalha sozinho, de forma isolada, um usuário ao retornar para uma tarefa em que estava trabalhando pode, facilmente, buscar em sua memória as mudanças significativas realizadas em uma sessão anterior, relembrando o que ele alterou nos objetos e o que ainda necessita ser feito. Por outro lado, ao trabalhar com outras pessoas, cooperativamente, ao retornar ao trabalho em um artefato compartilhado o

usuário pode encontrar mudanças ocorridas devidas à ação de outras pessoas. Nesse caso, ele não tem como adivinhar o que foi feito, o que mudou e o que ainda está por fazer. No entanto, ele necessita dessa informação para compreender as influências dessas mudanças sobre seu trabalho e como ele deverá agir desse ponto em diante.

Quando decisões e êxitos dependem da integração de diferentes membros de um grupo, torna-se importante que cada um conheça o progresso do trabalho dos companheiros, como o que falta para o término de suas atividades, quais os resultados preliminares, etc. Sem tal contexto, os indivíduos não podem medir a qualidade de seu próprio trabalho relativo aos objetivos e progressos do grupo (DOURISH e BELLOTTI, 1992).

As informações de percepção tornam a colaboração mais eficiente, permitindo aos participantes evitar a duplicidade de trabalho e reduzir os conflitos. Prover facilidades adequadas para a percepção é essencial para colaborações suportadas por computador efetivas, uma vez que na falta dessas informações o usuário é obrigado a supor o que os outros estão fazendo, causando perda de tempo em atividades não produtivas (PRAKASH et al., 1999).

A falta de percepção em um grupo gera diversos problemas, como redundância nas tarefas, inconsistências e contradições, acarretando em um trabalho sem coesão de idéias ou incompleto, prejudicando seu resultado em termos de qualidade e eficiência, podendo, até, em casos extremos, não atingir seus objetivos (PINHEIRO, 2001). Além disso, a falta de informações de percepção, principalmente nas interações assíncronas, pode gerar um estado de solidão e inércia nos membros do grupo, agravado pelo desconhecimento da importância de suas atividades para o grupo, de seus prazos ou do andamento do trabalho (PINHEIRO, 2001).

A percepção é importante, também, para apoiar a coordenação de atividades individuais ou coletivas, a comunicação informal e o uso de convenções (SOHLENKAMP, 1998). Comunicação informal é a interação oportunista, espontânea e não planejada entre as pessoas.

### II.3.2. Fatores que Influenciam a Percepção

A forma como as informações de percepção devem ser coletadas e apresentadas pode variar de acordo com alguns critérios, como o modo de interação e o papel desempenhado pelo usuário dentro do grupo.

#### II.3.2.1. Modo de interação: síncrono X assíncrono

A percepção é importante tanto nas interações síncronas quanto assíncronas. Entretanto, o tipo de informação de percepção e os mecanismos correspondentes são diferentes nas duas formas de interação, em diversos aspectos, como em relação à granularidade das informações tratadas, ao fornecimento dessas informações aos usuários finais e à durabilidade das mesmas (PINHEIRO, 2001).

Usuários interagindo de forma síncrona estão interessados em eventos que estejam ocorrendo sobre o espaço de trabalho compartilhado no momento atual, com notificação imediata dos mesmos. Essa percepção possibilita que os usuários tenham uma mesma visão da área de trabalho (WYSIWIS – *What You See Is What I See*) (GUTWIN e GREENBERG, 2002). Assim, os eventos estão relacionados a ações de movimentação dos usuários pelo espaço de trabalho, sua disponibilidade para interação, e ações de baixa granularidade como movimentações de *mouse* e cursor, alteração na posição da barra de rolagem, etc. As informações de percepção geradas para apoiar esse tipo de interação são efêmeras, necessitando baixa ou nenhuma persistência. Seu armazenamento é apenas na memória dos participantes da interação, ou então podem ser gravadas para reprodução posterior.

No modo de trabalho assíncrono, os participantes, geralmente, são informados sobre interações das quais os mesmos não tenham participado, em um momento posterior à ocorrência das mesmas. O suporte à percepção é derivado de uma interpretação resumida de uma seqüência completa de eventos, ocorridos no espaço de tempo (FUSSEL et al., 1995). As informações geradas devem estar em um nível mais macro, respondendo questões do tipo “quem está responsável por uma atividade”, “qual o prazo limite para o término da mesma”, ou um resumo das últimas ações executadas sobre um determinado artefato. Para possibilitar notificação posterior, as informações de percepção necessitam de alta persistência, pelo menos até que todos os indivíduos que tenham interesse nas mesmas tenham sido notificados.

Como os usuários em aplicações cooperativas costumam alternar momentos de interação síncrona e assíncrona, um mecanismo de percepção efetivo deve se preocupar com as questões de percepção relativas aos dois modos de interação.

### II.3.2.2. Papel desempenhado pelo participante no grupo

Um papel é um conjunto de privilégios e responsabilidades atribuídas a uma pessoa ou, algumas vezes, a um módulo de um sistema (ELLIS et al., 1991). Os participantes em uma equipe de trabalho, normalmente, possuem atribuições e responsabilidades diferenciadas, e assumem uma determinada posição hierárquica dentro do grupo.

De uma forma geral, analisando-se equipes de trabalho quaisquer, sem estabelecer um domínio específico, pode-se identificar, basicamente, três níveis hierárquicos de papéis que podem ser atribuídos aos usuários:

- **Nível operacional**: composto por usuários executores do trabalho final do grupo;
- **Nível tático**: formado por usuários com responsabilidade de acompanhar, controlar e gerenciar as atividades executadas pelos operadores, estimulando a interação, resolvendo possíveis conflitos e garantindo que o grupo siga uma metodologia de trabalho;
- **Nível estratégico**: ocupado por usuários com papel de definir metas e objetivos para o grupo, traçando processos e estratégias de ação, identificando, a partir do conhecimento sobre ações anteriores do grupo, boas práticas ou práticas que devam ser evitadas.

A necessidade de percepção de cada um desses usuários é bastante diferente. Operadores desejam receber informações relacionadas às suas próprias atividades e a artefatos de seu interesse. Assim, ele necessita saber, por exemplo, o que aconteceu aos artefatos nos quais estava trabalhando em períodos de ausência, quem modificou ou acessou os artefatos, etc. (PINHEIRO, 2001). É nesse nível que atuam a maioria dos mecanismos de percepção.

Por outro lado, os coordenadores não têm interesse em informações muito detalhadas sobre cada atividade executada por cada participante dentro do grupo. Esses usuários desejam visões de alto nível dos dados, através de resumos e agregações, de modo a reduzir a sobrecarga de informação e possibilitar reflexão sobre mudanças nas

ações, preferências, comportamento e performance do grupo (BORGES e PINO, 1999) (DAVID e BORGES, 2001). Através de uma distribuição gráfica das contribuições dos participantes, o coordenador pode identificar aqueles que estão trabalhando mais intensamente no processo. Taxas altas ou baixas de contribuição devem ser analisadas pelo coordenador, de modo a motivar ou inibir novas contribuições a determinados assuntos.

Já os analistas estratégicos atuam em um nível acima dos demais, numa posição de observador do que ocorre, em geral, sobre mais de um grupo de trabalho. Nas aplicações cooperativas pode-se alcançar esse tipo de percepção através de consultas analíticas sobre os dados históricos de interação do grupo.

### II.3.3. Formas de Prover Percepção

A maneira como a percepção pode ser provida passa por dois tópicos principais, discutidos a seguir: a forma como os usuários podem ser informados sobre mudanças efetuadas por outros usuários na área de trabalho e os modos de distribuição das informações que podem ser utilizados nos mecanismos de percepção.

#### II.3.3.1. Modos de comunicação: direta x indireta

Existem, basicamente, duas maneiras de manter os participantes atentos ao que ocorre no contexto de um grupo: a percepção ativa e a percepção passiva (DOURISH e BELLOTTI, 1992). A percepção ativa é alcançada quando os membros informam-se uns aos outros sobre o estado de suas atividades e planos, de uma forma consciente, como através de reuniões face-a-face, conversa nos corredores, correio eletrônico ou relatórios elaborados e enviados diretamente para os colegas.

No entanto, a tarefa "manual" de informar diretamente sobre atividades realizadas é bastante cansativa e sujeita a falhas e esquecimento. Ao final de uma sessão de trabalho, que pode durar até um dia inteiro, é difícil lembrar tudo o que foi realizado. E a noção do que é ou não importante varia muito de pessoa para pessoa. Com o passar do tempo a tendência é que haja desgaste no relacionamento dos membros da equipe, causando conflitos, mal entendidos, trabalhos duplicados ou perda de trabalho.

A percepção passiva é atingida de forma indireta, quando os membros não fazem um esforço consciente para a comunicação de suas atividades. Por exemplo, quando

uma pessoa ouve uma conversa por acaso e fica sabendo do que está acontecendo, ou quando é feito um monitoramento automático (com notificação posterior) das atividades desenvolvidas. Essa forma de percepção é mais difícil de ser alcançada, sendo o objetivo dos mecanismos de percepção. Uma forma de promover essa percepção é através da manipulação de artefatos compartilhados, onde o próprio artefato mantém o histórico das mudanças em seu estado interno, possibilitando que os usuários consultem esse histórico para compreender as mudanças ocorridas.

### II.3.3.2. Alternativas de Entrega das Informações de Percepção

Em seu livro sobre bancos de dados distribuídos, Özsu e Valduriez (ÖZSU e VALDURIEZ, 1999) apresentam o problema de entrega de dados em ambientes distribuídos, caracterizando-o em três dimensões ortogonais: modos de entrega, freqüência e métodos de comunicação. Como as aplicações cooperativas são, em geral, distribuídas, essa mesma caracterização pode ser aplicada ao problema de entrega das informações de percepção.

***Modo de entrega***

A entrega das informações de percepção pode ser realizada de três modos: puxe (*pull*), empurre (*push*) ou híbrido. No modo **puxe**, clientes obtém informações do servidor de percepção através de solicitações explícitas. A principal característica desse modo de entrega é que a chegada de novos dados ou atualizações de dados existentes são processadas pelo servidor, porém nenhuma notificação é disparada aos clientes, até que os mesmos solicitem esses dados. Um ponto ruim desse tipo de entrega é que os clientes devem, obrigatoriamente, conhecer quais informações desejam procurar e quando realizar essa procura. É o modo de entrega principal empregado pelos SGBDs.

No modo **empurre**, a entrega das informações do servidor para o cliente é executada por iniciativa do próprio servidor, sem que uma requisição explícita do cliente tenha sido efetuada. A principal dificuldade nesse modo de entrega é decidir quais dados interessam ao cliente e quando o envio deverá ocorrer (periodicamente, a qualquer momento ou sob alguma condição). Desse modo, a utilidade desse serviço de entrega é definida fortemente pela capacidade do servidor em prever as necessidades dos clientes. Um problema dessa abordagem é a possibilidade de sobrecarregar o usuário com informações que não sejam do seu interesse ou que o mesmo já tenha

conhecimento. Esse é o modo de entrega comumente empregado pela maioria dos mecanismos de percepção.

O modo de entrega **híbrido** combina os modos puxe e empurre. Uma forma de implementar esse modo de entrega, aplicado na abordagem de consultas contínuas (LIU et al., 1996), sugere que a primeira solicitação seja executada pelo cliente (puxe) e as transferências ou atualizações de informações posteriores sejam disparadas pelo servidor (empurre). Nesse modo, os clientes recebem, continuamente, do servidor, informações que casem com seu perfil. É uma abordagem que vem sendo adotada nos mecanismos de percepção para reduzir a sobrecarga de informações.

***Freqüência***

A freqüência de entrega da informação pode ser classificada segundo três medidas: periódica, condicional e irregular. Na entrega **periódica**, os dados são enviados do servidor para o cliente em intervalos regulares. Os intervalos podem ser definidos pelo sistema ou pelos próprios clientes (através do seu perfil). Na entrega **condicional** os dados são enviados dos servidores sempre que determinadas condições, indicadas pelos clientes em seus perfis, são satisfeitas. Na entrega **irregular,** os dados são enviados dos servidores para os clientes de modo *ad-hoc*, sempre que o cliente faz uma requisição.

***Método de Comunicação***

O terceiro item a ser analisado para efetuar entrega de informações é o método de comunicação, que determina os vários modos pelos quais servidores e clientes se comunicam para trocar informações. As alternativas são *unicast* (**um-para-um**) e um-para-muitos. No modo *unicast*, o servidor envia dados para um cliente utilizando um modo de entrega particular com alguma freqüência. No modo **um-para-muitos**, o servidor envia os dados para um número de clientes, usando um protocolo de comunicação de transmissão múltipla, simultânea.

## II.3.4. Problemas em Prover Percepção

Enquanto a interação entre pessoas em uma situação face-a-face parece natural, visto que sentidos como visão e audição estão disponíveis em sua plenitude, essa

interação fica menos clara quando há a tentativa de fornecer suporte à percepção em ambientes virtuais (ASSIS, 2000).

Existem muitas dificuldades em prover percepção em aplicações cooperativas, em relação às interações face-a-face (GUTWIN e GREENBERG, 2002):

- Os dispositivos de entrada e saída utilizados nessas aplicações geram apenas uma fração da informação de percepção disponível em interações face-a-face;
- A interação de um usuário com uma área de trabalho computacional gera menor quantidade de informação do que ações em uma área de trabalho física;
- As aplicações cooperativas, geralmente, não apresentam nem as informações de percepção limitadas que estão disponíveis no sistema.

Além das deficiências tecnológicas em reproduzir situações de interação do mundo real, dois problemas básicos estão associados com a provisão de informações de percepção: a violação de privacidade e a sobrecarga de informações (PRAKASH et al., 1999)(ARAÚJO, 2000)(LIECHTI, 2000).

### II.3.4.1. Violação de privacidade

A violação de privacidade ocorre devido à possibilidade de publicação indevida de detalhes de atividades dos usuários que deveriam ser mantidos em sigilo. Os mecanismos de percepção visam tornar pública a ação dos membros de uma equipe, porém essa exposição pode gerar uma reação negativa nas pessoas envolvidas, por acreditarem que estes mecanismos possam ser utilizados com outros objetivos, como para controlar as pessoas (ARAÚJO, 2000).

### II.3.4.2. Sobrecarga de Informações

A quantidade de informações de percepção produzidas durante as interações em uma aplicação cooperativa tende a crescer progressivamente, ao longo do tempo, podendo ocasionar sobrecarga de informações nos usuários, causando perturbações e desviando o foco de sua atenção devido à chegada de informações, muitas vezes irrelevantes e desnecessárias ou que já sejam do seu conhecimento.

Esse problema deve ser evitado pelos mecanismos de percepção, pois o excesso de informação dificulta a concentração dos usuários nos aspectos essenciais do seu trabalho. Além disso, informação demais é o mesmo que nenhuma informação.

Os mecanismos de percepção devem levar em consideração duas questões para evitar a sobrecarga (CADIZ et al., 1998): (i) que informações são mais valiosas para os usuários e quais podem ser ignoradas? (ii) como apresentar essas informações, quando disponíveis, sem distrair as pessoas do foco de suas tarefas?

Uma forma de diminuir ou evitar a sobrecarga é através da utilização de filtros na captura e na apresentação das informações de percepção e o uso de perfis para os usuários, selecionando eventos, objetos e componentes de percepção que os mesmos tenham interesse em monitorar (DAVID e BORGES, 2001) (PINHEIRO, 2001) (PRAKASH et al., 1999).

Visando diminuir a sobrecarga de informações apresentadas aos usuários, os mecanismos de percepção podem aplicar conceitos de percepção contextual e de percepção periférica.

***Percepção Contextual***

Contexto é qualquer informação que pode ser utilizada para caracterizar a situação de uma entidade, onde uma entidade é uma pessoa, lugar ou objeto considerado relevante para a interação entre um usuário e uma aplicação (DEY, 2001).

A percepção contextual utiliza o conceito de contextos para determinar que tipo de informação os usuários desejam estar atentos e de que forma eles podem estar atentos, de modo a separar informações de percepção das informações de "perturbação" (LIECHTI, 2000). Nesse tipo de percepção deve-se identificar quais são as pessoas do grupo e que artefatos eles estão interessados, de modo que somente um subconjunto dos eventos possa ser identificado como crítico e notificado ao usuário.

***Percepção Periférica***

A percepção periférica denota que as informações de percepção devem ser apresentadas aos usuários do grupo, de forma periférica, sem exigir o foco de sua atenção (LIECHTI, 2000), oferecendo visões rápidas, ou "pelo canto dos olhos", ou "por sobre as cabeças", que permitam aos usuários saber de uma forma pontual e sem esforço o que seus colegas estão fazendo (ARAÚJO, 2000).

### II.3.5. Classificação das Informações de Percepção

Em termos gerais, para compreender totalmente uma interação da qual faz parte, um indivíduo deve conhecer as pessoas que compõem o grupo do qual participa, deve ter uma visão geral das atividades desenvolvidas pelo grupo, compreendendo onde suas próprias atividades se inserem dentro do contexto global, e deve ser informado sobre mudanças que venham a ocorrer no espaço de trabalho compartilhado, no decorrer das interações do grupo.

Diversos trabalhos na literatura apresentam classificações para a percepção a partir da natureza da informação associada (GUTWIN e GREENBERG, 1996) (SOHLENKAMP, 1998) (PRINZ, 1999) (BÜRGER, 1999) (PINHEIRO, 2001) (LIECHTI, 2000). As próximas subseções descrevem um resumo da classificação apresentada em (ARAÚJO, 2000).

#### II.3.5.1. Percepção social

As informações de percepção social são aquelas que auxiliam o usuário a conhecer o grupo no qual está participando, oferecendo dados sobre o mesmo e sobre os demais participantes. Este conhecimento auxilia os participantes a estabelecer as conexões sociais necessárias para o bom andamento da interação (BÜRGER, 1999). Essas informações incluem:

- O conhecimento sobre a composição do grupo, quem são seus membros e seus respectivos papéis e responsabilidades;
- A localização geográfica de cada participante, auxiliando-os a manter uma noção de espaço e distribuição;
- A presença do indivíduo no grupo, identificando quem está participando de uma interação, no caso das aplicações síncronas, e quem executou determinadas ações no espaço de trabalho, para as aplicações assíncronas;
- O grau de proximidade entre os participantes de um trabalho coletivo, tanto em termos de distância física quanto em papéis e responsabilidades, de modo que os usuários possam identificar parceiros com tarefas próximas ou relacionadas, que devam ser contatados para o auxílio na resolução de problemas e conflitos e na tomada de decisões em relação a suas atividades;

- Conhecimento sobre a disponibilidade dos usuários, de modo a possibilitar a comunicação entre os membros do grupo;
- Visualização da reação emocional dos participantes a cada fato que ocorre na interação, estabelecendo noções de satisfação ou insatisfação em relação às contribuições geradas.

### II.3.5.2. Percepção de Atividades

Esse tipo de percepção está focado nas atividades desempenhadas para se atingir um trabalho específico e os objetivos do grupo. Os participantes devem ser contextualizados sobre os objetivos e a estruturação das atividades que compõem o trabalho do grupo e devem poder acompanhar o desenrolar das mesmas, ou seja, quais atividades foram finalizadas, quais ainda estão em andamento e quais estão por realizar, bem como problemas ocorridos na realização das mesmas, como conflitos.

Esse tipo de percepção visa evitar redundâncias na execução dos trabalhos, ou que atividades deixem de ser executadas, além de facilitar a coordenação das atividades entre os próprios membros do grupo. Sua importância é evidenciada, particularmente, em interações assíncronas, onde os participantes não trabalham simultaneamente, podendo o desconhecimento de suas atividades e de sua posição na estrutura de atividades do grupo incorrer em duplicidade de trabalho ou em alto nível de conflitos.

### II.3.5.3. Percepção do Espaço de Trabalho Compartilhado

As duas categorias de informações de percepção anteriores provêm contexto aos usuários que permite que os mesmos possam iniciar suas atividades dentro do grupo. Entretanto, à medida que interações vão ocorrendo no espaço de trabalho compartilhado é importante que os participantes tenham conhecimento sobre o desenrolar dessas interações, de modo que possam perceber ações que interfiram em suas atividades. O apoio a este tipo de percepção é o foco desta dissertação.

A percepção do espaço de trabalho compartilhado (w*orkspace awareness*) é definida como a compreensão, no momento atual, da interação de uma outra pessoa com uma área de trabalho compartilhada, envolvendo o conhecimento sobre onde outros estão trabalhando, o que estão fazendo e o que farão a seguir (GUTWIN e GREENBERG, 1996). As informações que apóiam esse tipo de percepção incluem:

- O estado dos artefatos compartilhados, permitindo identificar quais são os artefatos presentes no espaço compartilhado, qual o histórico de manipulação dos mesmos e qual o seu estado atual (concluídos ou em fase de construção);
- As ações realizadas pelos participantes dentro do espaço de trabalho, indicando quais foram essas ações, onde elas ocorreram, quem as realizou, quando, porque e como elas foram realizadas e qual o impacto causado sobre o espaço de trabalho;
- O posicionamento de cada participante da interação no espaço de trabalho, indicando qual o foco de atenção de cada membro do grupo e que objetos estão manipulando. Esse tipo de informação é mais crítico em interações síncronas.

Gutwin e Greenberg (GUTWIN e GREENBERG, 2002) destacam que a percepção do espaço de trabalho: auxilia as pessoas a notar e gerenciar transições entre trabalho individual e compartilhado; permite que essas pessoas utilizem a área de trabalho e os artefatos compartilhados como suporte conversacional, incluindo mecanismos de demonstrações e de evidência visual; auxilia a coordenação das ações, possibilitando às pessoas planejar e executar ações de baixo nível na área de trabalho para combinar suas atividades com os outros; permite às pessoas predizer ações e atividades dos outros em diversas escalas de tempo, antecipando-se a essas ações; e, auxilia a compreensão do contexto onde uma ajuda deve ser provida.

### II.3.6. Representação das Informações de Percepção – 5W + 1H

Eventos são pequenas mensagens estruturadas que contêm informações sobre mudanças de estado dos artefatos armazenados e que são propagadas pelo ambiente de modo a prover percepção (TREVOR et al., 1993). A maioria dos mecanismos de percepção baseia-se nas facilidades do modelo de eventos (FUSSEL et al., 1995; PINHEIRO et al., 2002; PRINZ, 1999; SOHLENKAMP et al., 2000).

A representação, conhecida como 5W+1H, identifica seis questões básicas que devem ser respondidas quando se deseja prover contexto a um indivíduo sobre algo que o mesmo não tenha conhecimento. São elas: quem (***Who***), o que (***What***), onde (***Where***), quando (***When***), porquê (***Why***) e como (***How***). Essas questões aparecem freqüentemente na literatura (DOURISH e BELLOTTI, 1992) (GUTWIN, 1997) (SOHLENKAMP, 1998) (MCCAFFREY e L., 1998), associada a diferentes contextos, não apenas percepção em aplicações cooperativas, sempre com a idéia de representar o conjunto

mínimo de informações necessárias para iniciar o usuário em algum tópico (*Questions to Help You Get Started*). Elas, inclusive, deram nome à trilha sobre CSCW do CHI2000: *The What, Who, Where, When, Why and How of Context-Awareness*.

A semântica de cada uma dessas perguntas, e o que elas se propõem a responder, pode variar a depender do contexto onde as mesmas são utilizadas. Em (PRESSMAN, 1999) elas são apresentadas (acrescidas da questão "quanto") como parte de um processo de desenvolvimento de *software*, sendo empregadas para guiar a definição das características chaves de um novo projeto e o plano resultante do mesmo, através de respostas às seguintes perguntas: "Porque o sistema está sendo desenvolvido?", "O que deverá ser feito e quando deverá ficar pronto?", "Quem é responsável por uma função?", "Onde essas pessoas estão localizadas dentro da organização?", "Como o trabalho será realizado?", "Quanto de recurso é necessário?".

Em (PINHEIRO, 2001) elas são utilizadas como guia para indicar considerações que devem ser analisadas ao se projetar um mecanismo de percepção. A questão "o que" refere-se a quais informações devem ser fornecidas aos usuários a depender do papel desempenhado pelo mesmo dentro do grupo. A questão "quando" trata do momento em que eventos ocorreram e devem ser propagados para os demais usuários. A questão "onde" indica em que local as informações são geradas e apresentadas. A questão "como" trata da maneira como as informações serão apresentadas aos usuários. A questão "quem" refere-se à noção de presença e disponibilidade das pessoas dentro do grupo. A questão "por que" não foi tratada, sendo substituída por "quanto", no sentido de indicar a quantidade de informação suficiente para promover percepção sem sobrecarregar o usuário.

Em (TAM, 2002), essas questões são utilizadas para prover percepção sobre mudanças ocorridas no espaço compartilhado e visam responder: "Quem é o responsável pela mudança?", "Onde as mudanças ocorreram?", "Que mudanças foram feitas?", "Quando as mudanças aconteceram?", "Porque a mudança foi feita?", "Como ocorreram as mudanças?".

Pode-se atribuir diferentes pesos a cada uma das perguntas a depender da visão do participante. Se o mesmo está interessado nas pessoas que compõem o grupo, ele estará mais concentrado na questão "quem", no entanto, se seu foco é mais nas mudanças sobre os artefatos que compõem o espaço compartilhado, ele deverá dar uma atenção maior às questões "o que" e "onde" (GUTWIN e GREENBERG, 2002). Segundo

Gutwin (GUTWIN, 1997), para qualquer ambiente de *groupware* o nível desejado do histórico de percepção provavelmente será uma combinação dessas seis questões.

### II.3.7. Percepção em SGBDs

Em aplicações cooperativas, a interação ocorre, geralmente, através do acesso e manipulação de informações e artefatos compartilhados (PREGUIÇA et al., 2000) (GREIF e SARIN, 1987), os quais, em geral, são mantidos persistentes através de SGBDs. Os membros do grupo podem explorar a base de informações como um meio de contexto compartilhado ou como um meio de coordenar suas atividades (GREIF e SARIN, 1987). Conhecer o que mudou na base de dados torna possível saber o que os usuários do grupo andaram fazendo no espaço compartilhado, além de fornecer um histórico das interações do grupo.

Mariani (MARIANI, 1997) caracteriza o compartilhamento de informações através da expressão:

**Compartilhamento de informação = sistemas multi-usuários + bloqueio identificado + percepção**

onde, bloqueio identificado refere-se a bloqueios que mantêm informações que possam ser apresentadas aos usuários relativas ao estado do bloqueio. Desse modo, ao invés de simplesmente saber que um recurso está livre ou bloqueado, para promover a percepção, o usuário pode ter conhecimento sobre quem mantém o bloqueio e o porquê dessa manutenção.

Nesse mesmo trabalho, Mariani define alguns requisitos que deveriam ser atendidos pelos SGBDs para servir aos mecanismos de percepção (MARIANI, 1997):

- **Repositório de dados reativo**: quando um item é modificado, essa mudança deve ser propagada aos usuários;
- **Gerência de eventos**: Cada ação que afete o SGBD produz um evento associado;
- **Níveis de percepção**: diferentes tipos de ações podem exigir tratamentos diferenciados de percepção;
- **Dados temporais**: informações históricas (temporais) são importantes, para habilitar a realização de consultas temporais, a recuperação de estados anteriores do objeto ou consultas a eventos ocorridos.

### II.3.7.1. Bancos de Dados Ativos

A área de pesquisa de Bancos de Dados Ativos (BDA) estuda formas de estender as capacidades dos SGBDs passivos com funcionalidades que os tornem capazes de responder a eventos gerados interna ou externamente ao sistema, testando determinadas condições e executando ações como conseqüência deste teste. Assim, quando um determinado evento ocorre no banco de dados e uma dada condição é atingida, o SGBD Ativo (SGBDA) responde com alguma ação programada pelo administrador do banco de dados.

Componentes fundamentais de um SGBDA são as regras ECA (Evento-Condição-Ação) que descrevem como o sistema deve se comportar. O evento indica a ocorrência de uma situação, subdividindo-se em três tipos: temporais (ocorrência em um determinado instante de tempo), definidos pelo usuário e operações próprias do BD. Já a condição é um predicado sobre o estado do banco de dados e a ação corresponde às operações a executar quando o evento ocorre e a condição for verdadeira.

Existem, basicamente, dois tipos de regras ECA nos SGBDAs: restrições (*constraints*), que geram alguma exceção quando uma determinada condição ocorre, e gatilhos (*triggers*), que iniciam alguma ação quando uma condição se torna verdadeira (CERI et al., 2000). Essas regras, geralmente, são propostas para manter a consistência da base de dados. Argumenta-se que esses mecanismos de consistência são melhor definidos no nível do dado do que nas aplicações que manipulam o dado (MARIANI, 1997). Desse modo, a idéia de promover percepção segue o mesmo argumento: como os objetos são compartilhados, poderiam ser utilizadas regras para sinalizar a atividade dos usuários, associando eventos do usuário a mudanças no banco de dados.

No entanto, simples mecanismos de bancos de dados ativos não são ideais para apoiar a percepção em aplicações cooperativas. Isso se deve por que para que os artefatos do SGBD sejam monitorados, é necessário que regras ECA sejam programadas diretamente sobre cada um dos artefatos, o que implica em um grande trabalho do administrador da base de dados e uma alta dependência do modelo de dados específico da aplicação. Mudanças no esquema desses artefatos impõem que as regras sejam recriadas. Além disso, para que a percepção seja promovida é necessário que um mecanismo receba as informações coletadas sobre as ações ocorridas sobre os artefatos, trate essas informações e as apresente aos demais usuários. O que não é possível apenas

utilizando as regras dos bancos de dados ativos. Assim, um mecanismo mais efetivo de percepção faz-se necessário.

### II.3.8. Mecanismos de Suporte à Percepção em Aplicações Cooperativas

Os mecanismos de percepção são as técnicas empregadas por um sistema para fornecer informações de percepção. O tipo de informação considerada e a granularidade das mesmas pode ser maior ou menor a depender do contexto em que o mecanismo está inserido.

Em (DIAS, 1998), são apresentadas quatro abordagens que devem ser analisadas quando da construção de mecanismos de suporte à percepção:

- **Recursos de interface**: uso de componentes de interface que podem ser facilmente integrados ao próprio ambiente de trabalho, facilitando a absorção das informações de percepção pelos usuários;
- **Consulta e navegação pela memória do grupo**: busca de informações sobre as interações e atividades de cada usuário referentes a determinados artefatos;
- **Anotações**: permitem que usuários possam inserir anotações onde registrem idéias, sugestões e comentários, especialmente úteis em interações assíncronas;
- **Notificações**: uso de notificações automáticas, que permitam que cada membro do grupo tome conhecimento de eventos ocorridos durante interações do grupo.

Essas abordagens podem ser combinadas e ampliadas na construção de um mecanismo de percepção, a depender das necessidades do sistema ao qual estará associado.

Como a percepção é um requisito comum a toda a classe de aplicações cooperativas, seria natural que as soluções existentes pudessem ser facilmente adaptadas e utilizadas em diferentes contextos. No entanto, em geral, os mecanismos são desenvolvidos fortemente acoplados a uma aplicação cooperativa específica. Esse acoplamento dificulta a reutilização das soluções de percepção obrigando que os projetistas tenham que recriar os mecanismos a cada nova aplicação desenvolvida.

De forma generalizada, os mecanismos de percepção podem ser classificados em três categorias principais:

- **Componentes visuais de percepção (*widgets*)**, os quais visam apoiar a apresentação das informações de percepção em diferentes aplicações;
- **Mecanismos de percepção de domínio específico**, esses mecanismos não se preocupam, apenas, em apresentar as informações de percepção, mas também com a coleta e distribuição dessas informações. Eles são construídos de forma bastante acoplada, para atender aos requisitos específicos de uma aplicação particular. Assim, têm a desvantagem de de não serem facilmente transportados para diferentes aplicações, funcionando, somente no contexto para o qual foram criados. Estes são os mecanismos de percepção mais comumente encontrados nas aplicações cooperativas;
- **Mecanismos de percepção de propósito genérico**, já os mecanismos de percepção de propósito genérico também atuam na coleta, distribuição e apresentação das informações de percepção. Porém, eles são projetados para atender diferentes aplicações, em diferentes contextos. Para isso, eles estabelecem requisitos genéricos de percepção.

As subseções a seguir apresentam alguns exemplos de componentes visuais de percepção e de mecanismos de percepção de propósito genérico. Os mecanismos de domínio específico não serão detalhados, uma vez que não constituem interesse desta dissertação. Alguns mecanismos de propósitos genéricos serão analisados em maiores detalhes, sendo destacados os pontos positivos desses mecanismos e onde os mesmos falham e poderiam ser melhorados.

### II.3.8.1. Componentes visuais de percepção

Os componentes visuais de percepção são componentes tipicamente de interface, voltados para apresentar a informação de percepção de uma maneira útil e agradável ao usuário. Em geral, eles estão associados a mecanismos que fazem a coleta e distribuição das informações (ou seja, de onde os dados apresentados são originados). São focados em necessidades específicas de percepção, ou seja, cada componente visa apresentar a informação de uma determinada maneira, com um objetivo específico. Esses componentes, em geral, podem ser reutilizados em diferentes aplicações. A seguir, apresentaremos alguns exemplos de diferentes tipos de componentes visuais de percepção.

Os dois componentes a seguir são usados em interações síncronas, especialmente para facilitar a edição colaborativa, fornecendo percepção sobre o espaço de trabalho:

- **Visão por radar (*radar view*)**: apresenta uma visão em alto nível do espaço de trabalho, e uma indicação da região em que cada participante está focado no momento. Um exemplo desse componente é apresentado na Figura 4;
- **Teleapontador (*telepointer*)**: também chamado de múltiplos cursores (ROSEMAN e GREENBERG, 1996), é uma forma simples e efetiva de comunicar gestos dos usuários. Através desse mecanismo, é possível visualizar a movimentação do mouse feita por colegas. A Figura 4 apresenta dois usuários interagindo sincronamente em um ambiente de diagramação. O cursor em forma de mão ao centro indica a movimentação do mouse de cada usuário.

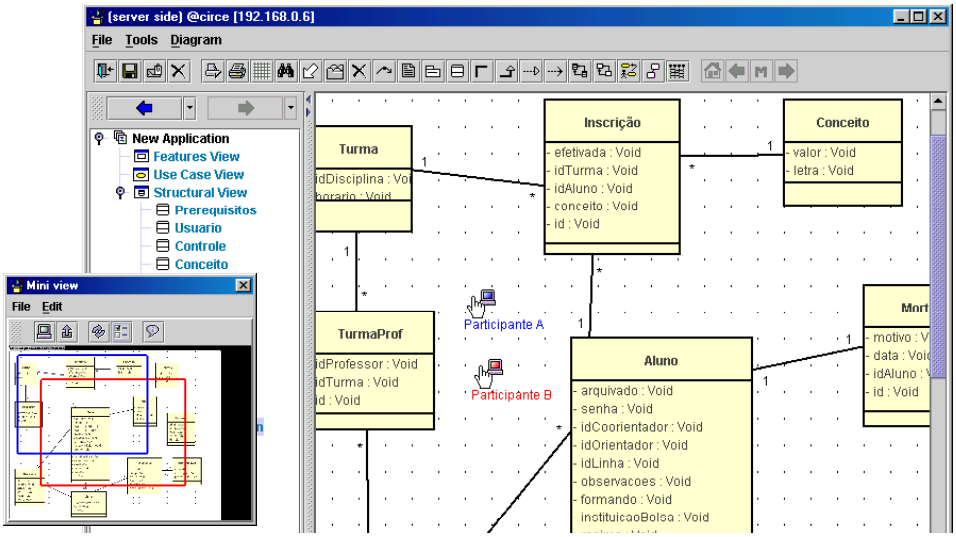



**Figura 4 - Componentes visuais de percepção no ambiente *OdysseyShare* - Teleapontadores (ícones ao centro) e visão de radar (janela em miniatura na esquerda) (MANGAN et al., 2002)**

Dentre os componentes visuais indicados para apoiar interações multi-síncronas, onde os usuários alternam momentos de interação síncrona e assíncrona, pode-se citar:

- ***Tickertape*** (PARSOWITH et al., 2003): ferramenta que exibe notificações de eventos sob a forma de mensagens de rolagem sobre uma janela de uma única linha. O *Tickertape* atua como um consumidor de eventos, recebendo informações de percepção de outras fontes. É indicado para exibir mensagens instantâneas ou eventos que tenham acabado de ocorrer. Além disso, ele persiste mensagens relevantes, permitindo que o usuário possa consultá-las posteriormente;

- **Histórico de Eventos**: este componente busca em um servidor de percepção pelo histórico de manipulação de determinados objetos, exibindo os eventos ocorridos de forma textual. Pode ser invocado diretamente pela aplicação (por exemplo, quando o usuário se conecta ao sistema), ou então ativamente pelo usuário. A Figura 5 apresenta uma interface do histórico de eventos utilizada no mecanismo POLIAwac (SOHLENKAMP, 1998).

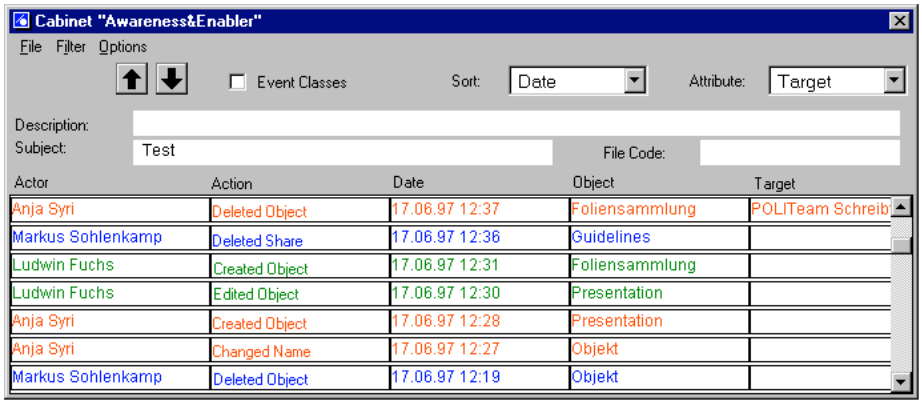



**Figura 5 - Histórico de Eventos - Mecanismo POLIAwac (SOHLENKAMP, 1998)**

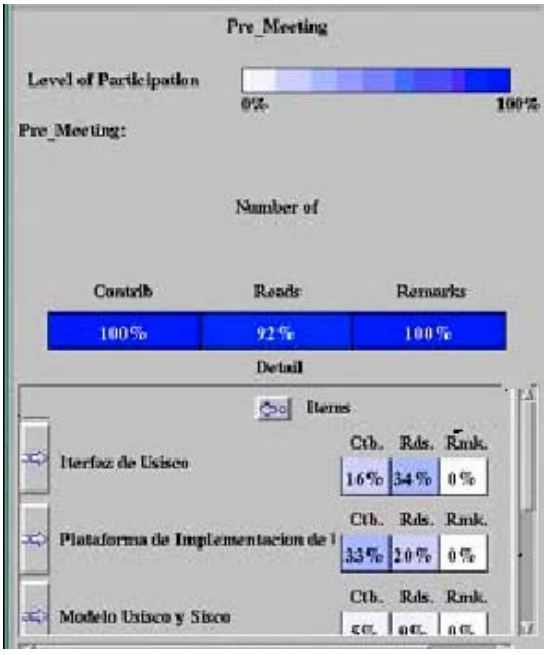



**Figura 6 – Participômetro (BORGES e PINO, 1999)**

Em (BORGES e PINO, 1999), são propostos alguns componentes para apoiar, explicitamente, usuários com papel de coordenador, apresentando as informações de forma agregada e resumida. Alguns desses componentes são:

- **Participômetro**: mecanismo (Figura 6) usado para mensurar, percentualmente, o nível de participação dos membros do grupo em relação à soma de participações;
- **Impactômetro**: mecanismo que tenta medir o impacto de certas idéias sobre o grupo a partir da quantidade de contribuições geradas a partir dessa idéia. Essas contribuições podem ser para o bem (quando as idéias são bem aceitas pelo grupo) ou para o mal (quando a idéia não é bem aceita e as contribuições, na verdade, são para criticar ou se posicionar contra a idéia);
- **Contributômetro**: indica o nível de contribuição dos participantes. Difere do participômetro, pois este último também considera como participação a leitura de contribuições de outros usuários.

Outros exemplos de componentes visuais de percepção são os componentes de percepção do grupo (GAW - *Group Awareness Widgets*) (KREIJNS e KIRSCHNER, 2001) e os medidores de percepção (*Awareness Gauges*) (REDMILES et al, 2002):

- **GAW**: ferramentas usadas para implementar diferentes tipos de percepção do grupo permitindo, ao mesmo tempo, que os usuários se comuniquem uns com os outros. Um exemplo de GAW é apresentado na Figura 7. O contexto considerado nesse GAW é "presença no ambiente colaborativo". Assim, cada evento de conexão é capturado contendo o identificador do membro do grupo, o momento da sua ocorrência e a duração da conexão. Os eventos são exibidos graficamente em uma janela que contém no eixo Y os usuários conectados e no eixo X o tempo de conexão. Esse tempo é representado por faixas em cinza, cujo tamanho indica a duração da conexão. É utilizada uma escala logarítmica que permite exibir os pontos próximos do momento atual ($t = 0$) de momentos mais distantes ($t = \infty$.). O uso da escala logarítmica permite dar maior atenção a eventos ocorridos mais recentemente. A comunicação com um usuário é possível clicando no botão contendo o nome do usuário;
- **Medidor de percepção**: permite aos usuários monitorar determinados aspectos chave do seu ambiente sem distraí-lo de sua tarefa principal. Por exemplo, o piloto de um avião pode olhar rapidamente para o medidor de altitude, observando a altitude atual e voltando novamente sua atenção para o ato de pilotar. De forma idêntica, um desenvolvedor de *software* pode se beneficiar dos medidores para observar as mudanças que estão ocorrendo no sistema, dinamicamente, durante

seu uso, podendo a partir daí tomar atitudes para problemas que vão sendo observados. A Figura 8 mostra dois exemplos de medidores de percepção: o primeiro medidor é um gráfico que exibe atividades ao longo do tempo, enfatizando períodos de atividade mais intenso; o segundo é uma barra de progresso que notifica o usuário quando um número desejado de ações tenha ocorrido.

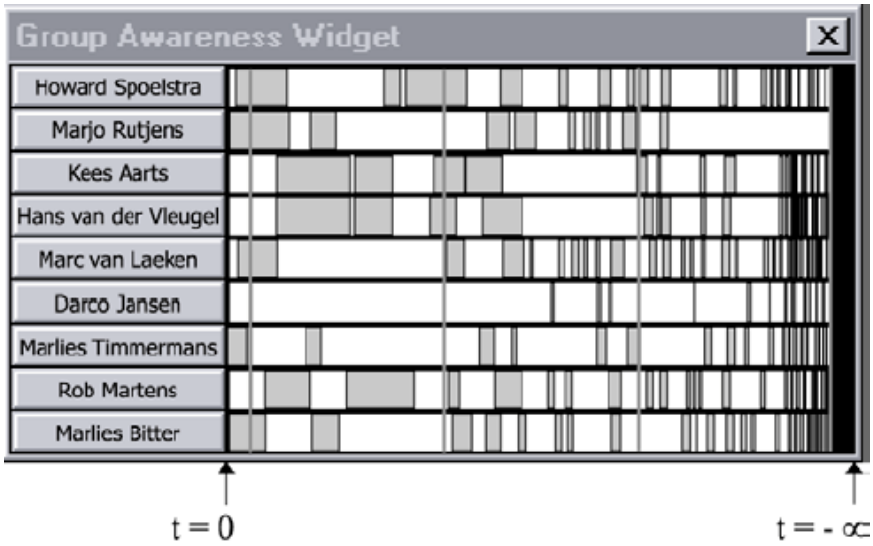



**Figura 7 – GAW no contexto "presença no ambiente" (KREIJNS e KIRSCHNER, 2001)**


**Figura 8 – Exemplos de medidores de percepção (REDMILES et al, 2002)**

### II.3.8.2. Mecanismos de percepção de propósito genérico

Uma abordagem promissora em mecanismos de percepção é a de infraestruturas de percepção genéricas e extensíveis que incluam mecanismos simples e leves para a geração e apresentação de notificações configuráveis aos usuários (PRINZ, 1999). Foram encontrados, na literatura, poucos mecanismos de percepção projetados para serem genéricos, atendendo a diferentes aplicações cooperativas, sendo esse tópico

considerado ainda em aberto. Dentre essas soluções, destacam-se o filtro de percepção SISCO (MARIANI, 1997), a infra-estrutura NESSIE (PRINZ, 1999) e o *framework* BW (PINHEIRO et al., 2002).

### *SISCO*

SISCO é um filtro de cooperação que provê percepção sobre acessos e mudanças em objetos armazenados em SGBD Orientado a Objetos (SGBDOO). Foi projetado como uma camada intermediária entre o cliente (aplicações que acessam o SGBD) e o servidor (o SGBDOO *Oggetto*). Toda e qualquer chamada do cliente deve passar antes pelo filtro, o qual fará o redirecionamento do pedido para o servidor. Esse filtro é usado, então, para capturar e armazenar todos os detalhes de acesso, os quais são persistidos em um repositório, sobrevivendo entre as várias sessões de trabalho dos usuários.

A implementação do SISCO foi realizada através de extensões ao SGBDOO *Oggetto*, restringindo sua aplicabilidade ao universo desse SGBD. O modelo de objetos do *Oggetto* foi estendido para se adequar ao conceito de área de trabalho, com a construção de coleções generalizadas: a definição da coleção foi estendida com uma lista de usuários e/ou papéis autorizados a acessar a coleção. As ferramentas do *Oggetto* foram modificadas para acessar o filtro SISCO, ao invés de acessar diretamente o servidor *Oggetto*, garantindo, assim, que todas as ações executadas por alguma aplicação sobre o *Oggetto* sejam monitoradas. E, foi construído um navegador colaborativo para exibir as informações de percepção produzidas.

Uma desvantagem dessa abordagem é sua forte dependência de um SGBDOO específico, restringindo-se a sistemas que utilizem esse SGBDOO. Na nossa proposta, estamos interessados em soluções generalizadas também no nível do SGBD, e não fortemente acopladas a um SGBD ou modelo de dados específico.

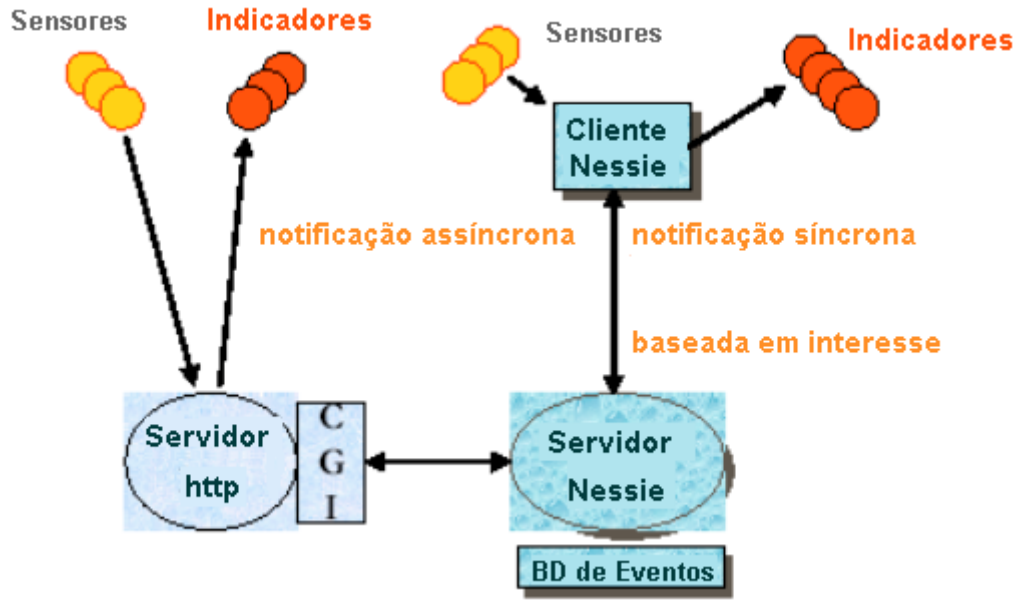
### *NESSIE*

Nessie (*awareNESS envIronmEnt*) é uma infra-estrutura genérica de percepção, independente de aplicação cooperativa específica, que inclui mecanismos para geração e apresentação de notificações configuráveis aos usuários. Uma questão de projeto estabelecida pelos pesquisadores do Nessie é que eventos relevantes sobre as atividades do usuário possam ser externalizados de uma aplicação e promovidos para outras

aplicações. Dessa maneira, o Nessie funciona como uma ponte para diferentes aplicações e configurações.

A Figura 9 apresenta uma visão geral da arquitetura do Nessie. As informações de cooperação para fins de percepção são capturadas através da geração de eventos. A recepção e a geração dos eventos são habilitados através da implementação de sensores, associados a atores ou artefatos compartilhados, que são responsáveis pela criação do evento. Após a criação, um evento é transmitido para o servidor de percepção o qual é responsável pela propagação, transformação e notificação dos eventos. Diferentes sensores podem ser usados para integração a outras aplicações.

Nessie utiliza o método baseado em perfis de interesse para especificar a informação de percepção que alguém está interessado. Estudos empíricos mostram que perfis de interesse constituem uma boa abordagem para a seleção de informações de percepção (SOHLENKAMP et al., 2000). A apresentação da informação é feita através de indicadores configuráveis. Ferramentas são oferecidas para permitir o mapeamento




de eventos de atividade a indicadores apropriados.

**Figura 9 – Visão Geral da Arquitetura da Infra-estrutura Nessie (PRINZ, 1999)**

O Nessie se concentra no apoio a eventos de ações relevantes que aumentam a percepção social e a percepção de tarefas no status de cooperação e não no status técnico da aplicação, ou seja, eventos do tipo movimentação do mouse não são tratados. Tais eventos ocorrem com freqüência menor e permitem outras arquiteturas e

demandam novas formas de apresentação da informação. Uma interface CGI/HTTP permite a comunicação entre aplicações e o servidor NESSIE, de modo que qualquer aplicação possa reportar eventos ou receber notificações do servidor.

A principal desvantagem dessa abordagem é que a aplicação deve se dirigir explicitamente ao servidor Nessie para informar a ocorrência de eventos ou para consumir novos eventos. Desse modo, os eventos são gerados através de chamadas explícitas da aplicação monitorada ou através de um cliente Nessie específico. Consideramos que os eventos deveriam ser gerados de forma automática, preferencialmente, sem onerar a aplicação cooperativa com mudanças ou tarefas adicionais para que os mesmos sejam coletados.


**Figura 10 – Ciclo Registro-Ocorrência-Notificação do *Framework* BW (PINHEIRO et al., 2002)**

### *BW (Big Watcher)*

O BW é um *framework* de apoio à percepção de eventos ocorridos no passado, voltado para interações assíncronas. BW utiliza percepção baseada em notificação de eventos executada segundo um ciclo em três fases: registro-monitoramento-notificação (Figura 10). Na primeira fase, a aplicação cooperativa registra no *framework* os tipos de eventos que têm interesse para os propósitos de percepção, passando, para isso, uma instância de exemplo de cada evento desejado para ser usado pelo BW. Na segunda fase, o usuário interage com a aplicação cooperativa para execução de atividades a qual deve monitorar essas atividades (aquelas que sejam de seu interesse para fins de percepção), gerar os eventos correspondentes a cada atividade e enviar esses eventos

para o BW. Na última fase do ciclo, os eventos ocorridos são repassados do BW para a aplicação cooperativa para que as informações de percepção sejam enviadas e apresentadas aos usuários finais. Para minimizar a sobrecarga de informação, o BW utiliza filtros baseados em perfis de interesse.

Alguns problemas foram observados nesse mecanismo: (i) por ser um *framework*, o BW não implementa todas as funcionalidades que podem ser necessárias para utilização do mecanismo pela aplicação, exigindo que extensões sejam feitas para que as aplicações utilizem seus serviços; (ii) a aplicação cooperativa precisa informar, explicitamente, sobre atividades ocorridas. Acreditamos que os eventos devam ser identificados e gerados pelo próprio mecanismo de percepção, sem que a aplicação necessite modificar suas funcionalidades para que o monitoramento ocorra.

## II.4. ANÁLISE DO APOIO À PERCEPÇÃO

Neste capítulo, vimos alguns conceitos relativos à área de CSCW e discutimos, especialmente, uma funcionalidade considerada muito importante para viabilizar a atividade cooperativa, a percepção. Discutimos os benefícios da percepção para viabilizar o trabalho em equipe, os problemas gerados com sua ausência, e os tipos de informações de percepção que podem ser tratadas pelos mecanismos de percepção (percepção social, percepção de atividades e percepção do espaço de trabalho). Apresentamos, também, como alguns autores consideram que os SGBDs poderiam apoiar a percepção, através da utilização de regras ativas.

Percebe-se que mecanismos efetivos de apoio à percepção ainda representam um problema nas aplicações cooperativas. A maioria dessas aplicações acaba implementando seu próprio mecanismo, fazendo pouca ou nenhuma reutilização de soluções de percepção existentes. Os mecanismos de percepção que se propõem a ser genéricos ainda apresentam alguns problemas, especialmente na questão do monitoramento automático de informações. Além disso, nenhuma das abordagens pesquisadas propõe o tratamento de valor agregado das informações de percepção históricas. Estas duas questões constituem-se, assim, os tópicos principais que este trabalho pretende explorar.

# *Capítulo III*

## *O Mecanismo de Percepção Ariane*

*Este capítulo apresenta a proposta do mecanismo de percepção Ariane, indica alguns requisitos impostos a este mecanismo, descreve o processo de percepção adotado, as principais funcionalidades do mecanismo e a arquitetura proposta.*

### III.1. INTRODUÇÃO

As aplicações de negócio, cooperativas ou não, em geral, utilizam SGBDs para armazenamento persistente de suas informações, possibilitando que os usuários tenham uma mesma visão dos dados manipulados pelas várias instâncias da aplicação. As ações dos usuários sobre objetos da aplicação provocam conseqüentes mudanças no estado interno dos artefatos armazenados no SGBD. No entanto, os SGBDs não implementam mecanismos que apóiem a percepção.

Os SGBDs, evidentemente, possuem conhecimento sobre mudanças ocorridas nos artefatos armazenados. Entretanto, esses sistemas, em geral, utilizam esses dados apenas internamente, como, por exemplo, para gerar *logs* que auxiliem em tarefas como controle de consistência. Seria bastante interessante para as aplicações cooperativas que esses sistemas adotassem uma atitude mais ativa, no sentido de não apenas registrar em *log* as mudanças ocorridas, mas que, por exemplo, comunicassem a ocorrência dessas mudanças a aplicações interessadas.

Quando as pessoas trabalham em equipe, de forma colaborativa, podem ser visualizados dois diferentes cenários de colaboração: um cenário onde os usuários trabalhem de forma isolada, e um outro onde os usuários trabalham percebendo as ações dos colegas. A Figura 11 e a Figura 12 ilustram essas duas situações de trabalho. Em ambas, dois usuários A e B estão interagindo sobre instâncias de uma mesma aplicação, persistindo informações em um banco de dados BD.

No primeiro cenário (Figura 11), o usuário A não tem conhecimento algum sobre as mudanças provocadas pelo usuário B sobre a base de dados e vice-versa. A seta

contínua indica as ações que estão sendo executadas pela aplicação sobre o BD. A direção da seta indica a origem da ação. Em um momento *t1*, o usuário A removeu da base de dados um artefato X, no qual o usuário B estava trabalhando anteriormente. Ao retornar em um momento *t2*, o usuário B não encontra mais o artefato X na base de dados. Como ele não foi notificado sobre as mudanças executadas por A, ele fica perdido, sem saber como agir desse ponto em diante.

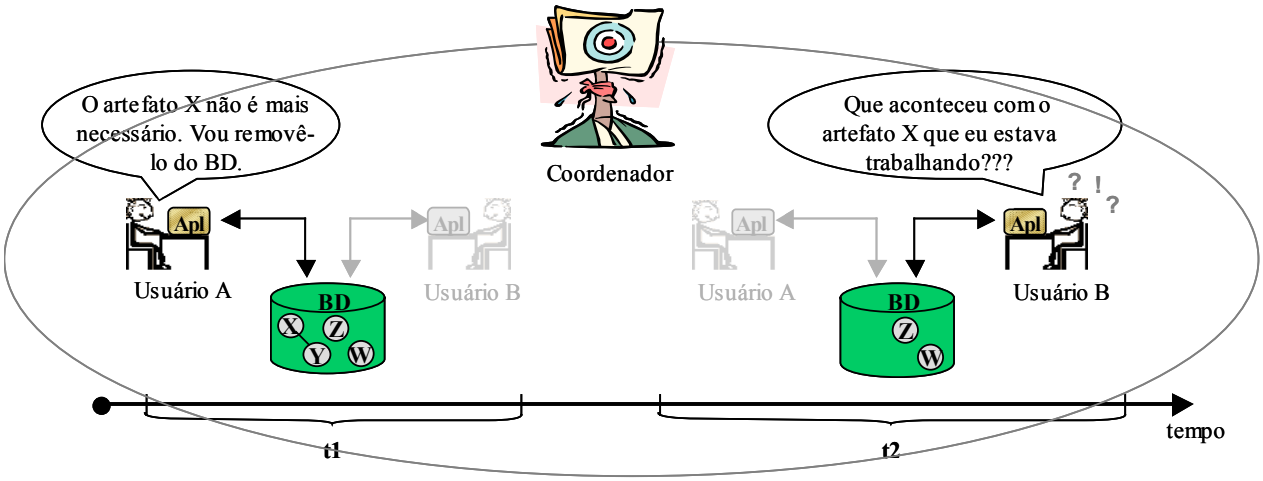



**Figura 11 – Interações com isolamento sobre uma base de dados compartilhada**

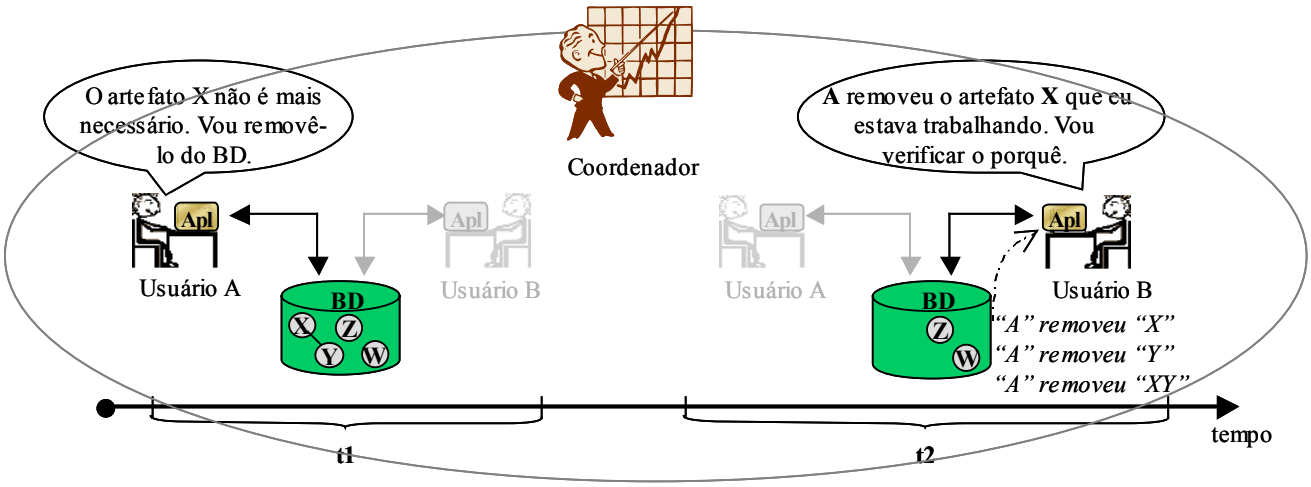



**Figura 12 – Interações com percepção sobre uma base de dados compartilhada**

Apesar de acessar e modificar uma mesma base de dados, os usuários não têm consciência das modificações ou consultas efetuadas pelos outros. Eles trabalham de forma isolada e independente. Caso ocorram conflitos, com mais de um usuário

tentando modificar o mesmo artefato, os mesmos são resolvidos pelo próprio SGBD, através de mecanismos de controle de concorrência e os usuários sequer tomam conhecimento, perdendo-se aí uma excelente oportunidade de interação entre indivíduos com interesses em artefatos comuns. Esse isolamento pode gerar diversos problemas como inconsistências e contradições e pode provocar desânimo e desmotivação nos membros da equipe. Além disso, a falta de informações sobre o que está acontecendo no contexto do grupo dificulta o trabalho dos usuários coordenadores no gerenciamento de suas equipe. Conseqüentemente, o grupo como um todo tem dificuldades em atingir seus objetivos.

No segundo cenário (Figura 12), o SGBD foi estendido com um serviço de percepção. Novamente, no instante *t1*, o usuário A remove o artefato X da base de dados. Como o artefato X estava relacionado a um outro artefato, Y, os dois artefatos foram removidos da base (inclusive o artefato relacionamento de X com Y). No instante *t2*, quando o usuário B retorna a sua área de trabalho, ele recebe uma notificação, enviada para a sua aplicação pelo BD estendido (indicado pela seta pontilhada), informando-o sobre as mudanças recentes ocorridas na base de dados, provocadas pelas ações de A. Conhecendo o que mudou na base de dados, ele pôde perceber que o artefato em que estava trabalhando foi removido pelo usuário A, possibilitando que ele tomasse a ação de conversar com A para entender o que o motivou a executar tal ação. Com isso, ele sabe a quem se dirigir para discutir e planejar suas próximas ações. Nesse cenário, os usuários conhecem e compreendem os eventos que ocorrem no âmbito do grupo. A percepção facilita a auto-coordenação e a resolução de conflitos, além de auxiliar os coordenadores no acompanhamento das atividades do grupo.

É a construção deste segundo cenário que esta dissertação pretende apoiar. Para isso, propomos um mecanismo de percepção que atue sobre o processo de persistência das aplicações cooperativas, monitorando as ações executadas por usuários dessas aplicações sobre os respectivos SGBDs. Com isso, espera-se ampliar a oferta de percepção dessas aplicações, sem implicar em custo adicional para elas, aproveitando-se, apenas, de um processo que ela já executa, normalmente, que é a persistência dos seus artefatos.

As próximas seções irão apresentar os aspectos gerais desse mecanismo de percepção, que foi denominado Ariane.

## III.2. REQUISITOS E RESTRIÇÕES

Alguns requisitos e restrições foram estabelecidos para o projeto de Ariane, de modo a delimitar o escopo da proposta, uma vez que o problema de percepção em aplicações cooperativas é bastante complexo. Os requisitos estabelecidos foram:

- **Independência**: Ariane não deve estar acoplado, internamente, a uma aplicação ou SGBD particular. A idéia é que possa ser reutilizado por diferentes aplicações, manipulando diferentes SGBDs. Deve-se reduzir, ao mínimo necessário, a intrusão de código na aplicação ou no SGBD utilizado, ou essa intrusão deve ser transparente para o desenvolvedor da aplicação;
- **Flexibilidade**: Espera-se que o mecanismo seja flexível, tanto do ponto de vista da aplicação, acompanhando a evolução da aplicação e mantendo sua funcionalidade, quanto à sua própria evolução, permitindo que funcionalidades adicionais sejam embutidas;
- **Manutenção da memória do grupo**: De modo a apoiar interações assíncronas onde os usuários podem ausentar-se por um longo período de tempo, a memória das interações ocorridas no grupo deve ser mantida;
- **Possibilitar análises sobre a memória do grupo**: Sobre a memória do grupo deve ser possível efetuar consultas e análises das interações ocorridas, utilizando ferramentas de consultas analíticas e/ou mecanismos de mineração de dados;
- **Suporte a diferentes categorias de usuários**: os participantes de um grupo possuem diferentes requisitos de percepção, a depender do papel que estejam desempenhando. O mecanismo deve prever e prover facilidades de manipulação dos dados, pensando nesses diferentes papéis.

Algumas restrições foram impostas ao mecanismo, em relação ao tipo de informação considerada e à apresentação dessas informações:

- **Tipo de Informação**: As informações consideradas são referentes ao conjunto de ações que podem ser executadas por aplicações sobre SGBDs, para manipular seus artefatos persistentes;
- **Apresentação da Informação**: o usuário final de Ariane não é diretamente um indivíduo, mas uma aplicação sobre a qual o indivíduo atua. Assim, a preocupação

maior de Ariane está em gerar informações de percepção e facilitar consultas sobre essas informações. Questões específicas, relacionadas à forma de apresentação dessas informações aos usuários, deverão ser tratadas pela aplicação que estiver utilizando o mecanismo.

## III.3. PROCESSO DE PERCEPÇÃO ADOTADO EM ARIANE

Segundo Sohlenkamp (SOHLENKAMP, 1998), o provimento de informações de percepção consiste de três etapas separadas e indispensáveis: a **coleta** das ações executadas sobre artefatos monitorados, a **distribuição** da informação de percepção associada a essas ações, sob a forma de eventos, e a sua **apresentação** a usuários interessados. Filtros devem ser aplicados em cada uma dessas etapas de modo a evitar sobrecarga de informações, pois nem toda ação deve ser monitorada, e nem todas as informações de percepção devem ser distribuídas ou apresentadas.

O processo de percepção adotado em Ariane, é uma adaptação do proposto por Sohlenkamp (SOHLENKAMP, 1998). A principal diferença entre eles é que em Ariane, estamos considerando mais duas etapas: o armazenamento das informações de percepção e a preparação das informações de percepção armazenadas para fins de análise. Imaginamos que técnicas de mineração de dados ou de processamento analítico (OLAP – *On-Line Analytical Processing*) podem ser empregadas para apoiar a análise das atividades e interações ocorridas no grupo.

A abordagem adotada por Ariane é a de percepção baseada em notificação de eventos. Dessa forma, o processo de percepção de Ariane está separado em 4 etapas principais: a produção, a distribuição, o consumo e a análise de eventos (Figura 13).

A ***produção*** dos eventos ocorre através da execução dos seguintes passos:

(i) **Captura de ação sobre o SGBD**: as ações executadas por usuários (produtores dos eventos), através das aplicações cooperativas, sobre os artefatos armazenados no SGBD são monitoradas e capturadas. A granularidade considerada para essas ações são as operações que podem ser executadas por uma aplicação sobre um SGBD;

(ii) **Geração de Evento**: das ações capturadas são extraídas informações relevantes para compreensão da mesma, gerando um evento estruturado segundo a

representação 5W+1H (explicada na seção **II.3.6**). Essa estruturação permite que os eventos sejam compreendidos e possam ser consumidos por diferentes componentes de apresentação, além de possibilitar a reconstrução da ação correspondente;

(iii) **Armazenamento**: os eventos gerados são armazenados em um repositório de eventos, que mantém o registro histórico de todas as ações ocorridas na base de dados, permitindo posterior consulta e recuperação dessas informações.

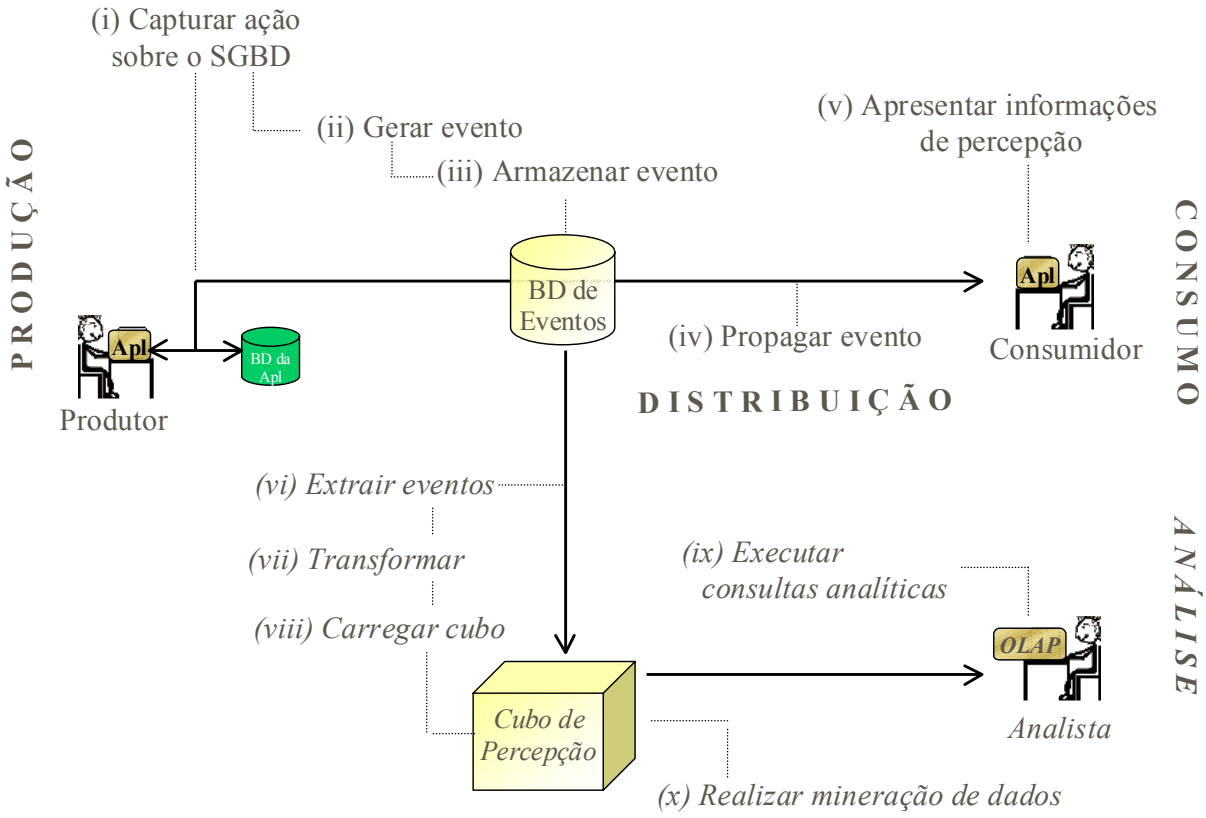



**Figura 13 – Processo de Percepção de Ariane**

A ***distribuição*** ocorre com:

(iv) **Propagação dos Eventos**: os eventos gerados são transmitidos e distribuídos a componentes visuais de percepção que tenham registrado interesse nos mesmos;

O ***consumo*** dos eventos se dá com a:

(v) **Apresentação**: componentes visuais de percepção tratam e exibem as informações de percepção contidas nos eventos, para os usuários consumidores. Essa

apresentação pode ocorrer de diferentes maneiras, sendo função do componente visual identificar a melhor maneira de tratar essa informação, considerando os requisitos do usuário que irá consumir essas informações;

Por fim, a etapa de análise ocorre com a preparação dos dados gerados pelo processo de percepção (eventos) e carga desses dados em uma estrutura apropriada para o processamento analítico e a mineração de dados, que denominamos Cubo de Percepção. Esse cubo é um banco de dados multidimensional. A modelagem multidimensional é um formato comumente utilizado para análise de dados, sendo amplamente empregado em sistemas OLAP (sistemas de apoio à tomada de decisão baseado em análise de alto volume de dados).

Para gerar o cubo de eventos pode ser empregado o processo Extração – Transformação – Carga (ETL - Extract-*Transform*-*Load*), bastante usual na carga de dados em armazém de dados (KIMBALL e MERZ, 2000). Como o próprio nome sugere, esse processo compõe-se de três etapas:

(vi) **Extração**: as informações contidas nos eventos gerados e informações complementares, de fontes externas, são extraídas e mantidas em uma área temporária;

(vii) **Transformação**: os dados são transformados para adequar-se ao modelo multidimensional, removendo-se informações desnecessárias e incluindo-se informações que aumentem a semântica dos dados;

(viii) **Carga**: os dados, transformados, são carregados no cubo de percepção;

Sobre o cubo de percepção criado podem ser executadas:

(ix) **Consultas Analíticas**: técnicas de processamento analítico (OLAP) podem ser utilizadas contra o cubo de percepção, de maneira que usuários do grupo com perfil de analista possam realizar consultas analíticas sobre as interações do grupo, compreendendo o comportamento dos participantes e analisando as atividades executadas. Essas análises podem auxiliar a tomada de decisões sobre a condução dos trabalhos do grupo;

(x) **Mineração de Dados**: "os sistemas OLAP emitem respostas para perguntas do tipo *que dados se encaixam nesse padrão?*, enquanto os sistemas de mineração de dados respondem à pergunta *que padrões existem nestes dados?*" (SULAIMAN,

2002) *apud* (CARVALHO et. al., 2001). Técnicas de mineração de dados podem ser aplicadas ao cubo de percepção visando descobrir conhecimento sobre as interações do grupo.

### III.3.1. Representação dos eventos

Os eventos são gerados segundo a estrutura 5W+1H descrita na seção II.3.6. Assim, cada evento deve guardar a informação sobre quem (***Who***) realizou qual operação (***What***) sobre qual artefato do SGBD (***Where***), em que momento essa operação foi executada (***When***), dentro de qual contexto (***How***) e qual o motivo para execução da mesma (***Why***). As 5 primeiras questões são respondidas diretamente pelas informações coletadas pelo sensor.

A questão *Why*, embora seja uma das mais importantes no processo de percepção, pois permite aos usuários conhecer os motivos por trás da ação de um usuário, auxiliando-os a compreender e aceitar essas mudanças, é tratada em Ariane, apenas para geração de eventos de exceção. Nesse caso, o por quê da exceção ter sido levantada é indicado pela mensagem de erro retornada pelo mecanismo de persistência. Para as demais operações (sessão, transação, atualização e consulta) essa questão não é tratada. Isso se deve porque o motivo pelo qual uma operação foi executada, em geral, está muito relacionado a aspectos internos do indivíduo executor da ação. As interações entre a aplicação cooperativa e o SGBD não trazem dados suficientes para responder a essa pergunta automaticamente. Pode-se indicar algumas relações entre as ações do tipo: antes de executar uma operação de remoção sobre um artefato o usuário irá executar uma operação de consulta sobre esse artefato. Assim, o motivo de consultá-lo é porque se deseja removê-lo.

Usualmente, os mecanismos de percepção solicitam que os usuários informem, manualmente, os motivos para a realização de uma ação (TAM, 2002). Entretanto, isso implicaria em mudanças no processo de persistência da aplicação cooperativa, pois o usuário seria obrigado a sempre que solicitar uma ação sobre algum artefato persistente ter que informar o motivo de estar fazendo isso, além de provocar a distração do usuário de sua tarefa atual, para prover essa explicação. Além disso, métodos manuais não são ideais para documentar mudanças pois podem ser incompletos, incorretos ou até omitidos (TAM, 2002).

Devido à dificuldade em documentar automaticamente essa pergunta e aos problemas em documentá-la manualmente, optou-se por manter essa questão na estrutura do evento apenas para tratar eventos de exceção.

### III.3.2. Filtros

A direção das setas da Figura 13 indica o fluxo da informação. Filtros devem ser fornecidos nas diversas etapas desse processo de modo a diminuir a sobrecarga de informações. Na etapa de produção de eventos, o filtro empregado é o tipo de informação considerada. Nem todas as ações ocorridas sobre a aplicação são tratadas, apenas as relativas a mudanças na base de dados. Com isso, tem-se um volume bem menor de informações com granularidade mais alta, do que se fossem tratadas todas as ações produzidas pela aplicação. Não foram considerados outros filtros para a produção de eventos, pois o que se deseja em Ariane é realmente monitorar tudo o que é produzido pela aplicação sobre o SGBD, para que todo o histórico seqüencial das ações fique armazenado. Filtros e agregações dos dados devem ser considerados nas etapas posteriores.

A etapa de distribuição dos eventos prevê a utilização de filtro baseado no tipo do evento. Os eventos são classificados pelo tipo de operação que realizam no SGBD (ex. atualização, consulta, etc). Os componentes de apresentação registram seu interesse em determinados tipos de eventos e, são notificados, apenas, quando eventos desse tipo ocorrerem.

Os eventos persistidos no repositório de eventos podem ser consultados utilizando alguns critérios de seleção, como: eventos ocorridos em um determinado período, eventos referentes a ações sobre determinados artefatos, eventos produzidos por um usuário específico, ou eventos que indiquem uma determinada operação.

Na etapa de consumo, quando as informações são apresentadas aos usuários finais, os componentes de apresentação devem implementar filtros adicionais, baseados no tipo de informação que interessa ao usuário final. Por exemplo, as informações podem ser filtradas pelos artefatos afetados, pelo tipo de operação ou por um determinado período de interesse do usuário.

Os sistemas OLAP incluem mecanismos de filtragens e agregações dos dados dentre suas funcionalidades. Assim, o cubo de percepção mantém o registro dos eventos ocorridos e o filtro é realizado na execução das consultas OLAP.

## III.4. PRINCIPAIS FUNCIONALIDADES DE ARIANE

A partir das atividades exibidas no processo de percepção de Ariane, pode-se identificar algumas funcionalidades principais do mecanismo. Essas funcionalidades podem ser resumidas através dos casos de uso gerais do mecanismo (Figura 14). Os atores, ou seja, as entidades externas que utilizarão informações e serviços do mecanismo, são: (i) a API monitorada, que representa a interface de comunicação entre a aplicação e o SGBD, de onde as ações serão capturadas; (ii) o banco de dados de eventos, onde os eventos produzidos serão armazenados; (iii) os componentes visuais de percepção, que apresentarão os eventos produzidos; e (iv) o cubo de percepção, que manterá a versão multidimensional dos eventos.

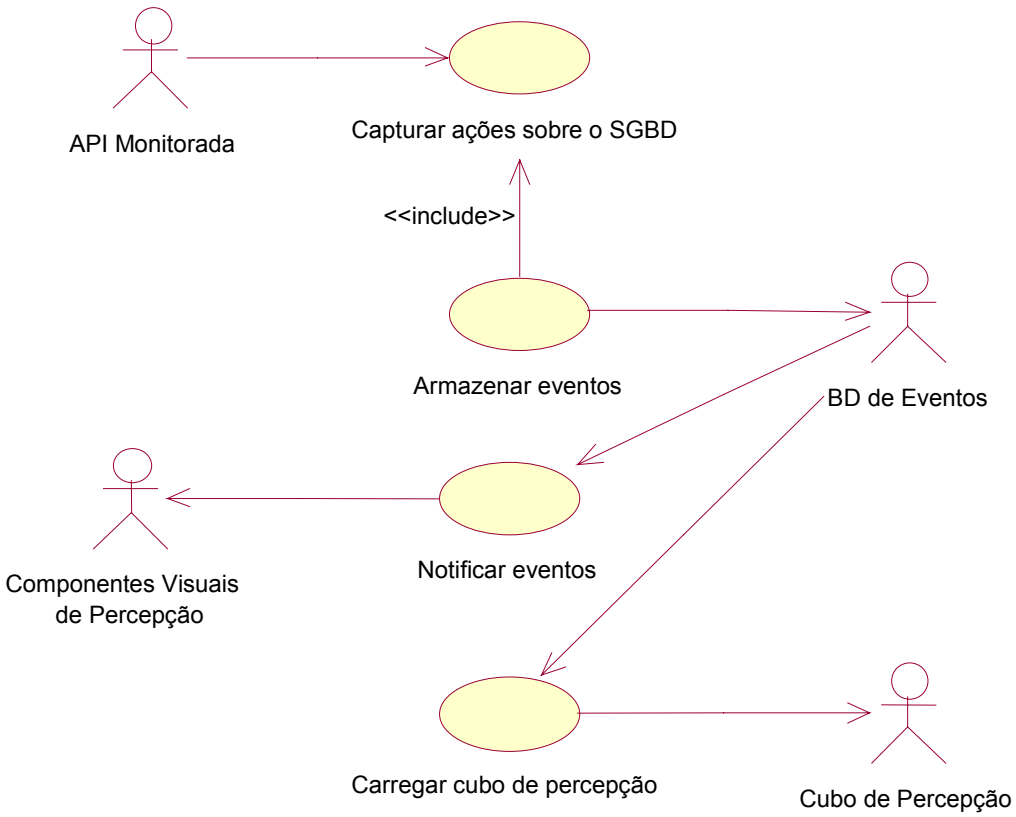



**Figura 14 – Diagrama de casos de uso com as principais funcionalidades de Ariane**

As funcionalidades principais de Ariane são, dessa maneira: (i) capturar ações ocorridas sobre o SGBD. Nesse caso, os tipos de ações consideradas são operações de conexão e desconexão ao banco de dados, transações e operações CRUD (criação, remoção, consulta e atualização) executada sobre os artefatos; (ii) armazenar as ações capturadas no BD de eventos, seguindo a representação 5W+1H; (iii) notificar os eventos a componentes visuais interessados; e, (iv) carregar os eventos do BD de eventos, modelados de forma multidimensional, no cubo de percepção.

## III.5. ARQUITETURA DE ARIANE

A Figura 15 ilustra a arquitetura de Ariane. Os componentes de Ariane (representados pelas elipses) foram separados em duas camadas: cliente e servidor. Essa divisão arquitetural é interessante para dissociar o serviço de percepção da aplicação que está sendo monitorada ou dos componentes de percepção que irão utilizar o serviço. Dessa maneira, a aplicação monitorada pode estar em uma máquina, acessando um SGBD localizado em uma segunda máquina e utilizando o serviço de percepção, que por sua vez está situado em uma terceira máquina. Ou, essa divisão pode ser apenas lógica, estando todos fisicamente em um mesmo local.

O funcionamento geral da arquitetura é o seguinte: Operadores interagem com instâncias de uma aplicação cooperativa a qual utiliza um mecanismo de persistência para armazenar seus artefatos em uma base de dados (BD da aplicação). Um **sensor** é acoplado à API de comunicação entre a aplicação e o SGBD, estendendo as funcionalidades da mesma, e possibilitando que as ações executadas sobre o SGBD sejam capturadas. As informações capturadas pelo sensor são encaminhadas ao **servidor de percepção**, onde essas informações gerarão eventos, que serão armazenados no repositório de eventos (BD de Eventos) e serão distribuídos para componentes visuais de percepção que tenham se registrado ao servidor de percepção como "ouvintes" dos eventos. Um processador ETL prepara e carrega os dados do BD de Eventos para o Cubo de Percepção, o qual é utilizado por usuários analistas (ou por coordenadores) do grupo, através de ferramentas OLAP, para a realização de consultas e análises.

As próximas seções definem e caracterizam cada um dos componentes da arquitetura de Ariane. Esses componentes foram classificados como elementos externos

que interagem com Ariane (os quais aparecem na arquitetura do lado de fora dos quadrados), e elementos internos de Ariane (que aparecem limitados pelos quadrados). Os componentes visuais, estão sendo representados como elementos internos de Ariane, por considerarmos que esses componentes são parte fundamental de um mecanismo de percepção, embora, como comentado anteriormente, a implementação desses componentes não faça parte do escopo desta dissertação.

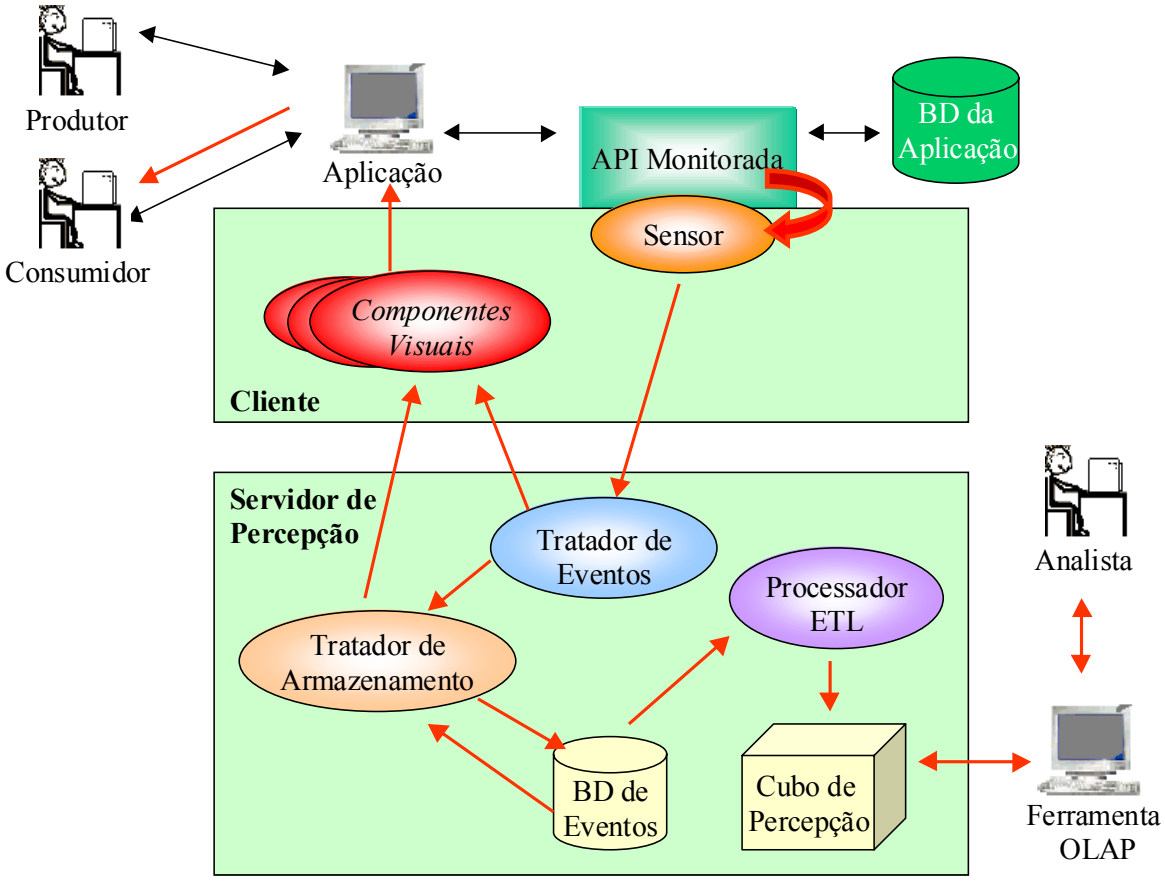



**Figura 15 – Arquitetura de Ariane**

### III.5.1. Elementos Externos que interagem com Ariane

- **Aplicação**: consiste da aplicação que deseja utilizar os serviços de percepção de Ariane. Teoricamente, essa aplicação seria uma ferramenta que apóie a atividade de equipes, mas essa não é uma regra, pois o mecanismo pode ser utilizado por qualquer aplicação interessada em monitorar seu processo de persistência;

- **BD da Aplicação**: Representa a base de dados utilizada pela aplicação para persistir seus artefatos. Essa base de dados pode estar sendo gerenciada por um SGBD qualquer (relacional, orientado a objetos, etc.);
- **API monitorada**: indica a interface de comunicação entre a aplicação e a base de dados que permite que a aplicação manipule seus artefatos armazenados;
- **Ferramenta OLAP**: representa um sistema de análise e suporte à decisão, que será utilizado para a realização de consultas e análises sobre o cubo de percepção.

## III.5.2. Elementos Internos de Ariane

Essa seção define e caracteriza os elementos internos de Ariane, discutindo o papel e responsabilidades de cada elemento para o funcionamento geral do mecanismo.

### III.5.2.1. Sensores

Os sensores são pequenos componentes utilizados para identificar, monitorar e capturar a ocorrência de determinadas ações da aplicação cooperativa sobre o SGBD. Eles trazem embutidos os tipos de operações sobre o SGBD que desejam monitorar e como as mesmas ocorrem no contexto que estão monitorando. Sempre que for executada uma operação do tipo definido, informações sobre a mesma são coletadas no sensor e enviadas para o servidor de percepção, que fará a geração do evento correspondente e dará seguimento ao processo de percepção. Essas informações indicam qual foi a operação executada, quem a executou, em que momento, que artefato do SGBD foi afetado pela operação e qual o contexto de execução da operação (por exemplo, uma identificação do mecanismo de persistência utilizado ou da rotina acionada nesse mecanismo para que a operação fosse executada.

As operações monitoradas pelo Sensor foram classificadas em 5 tipos:

- **Sessão**: trata das operações de **conexão** e **desconexão** a uma base de dados, indicando que a aplicação cooperativa iniciou ou finalizou uma nova sessão de trabalho com o SGBD;
- **Transação**: as operações executadas sobre um SGBD, em geral, ocorrem dentro de uma transação. A transação corresponde a uma unidade de trabalho, contendo uma operação ou um conjunto de operações, que devem ser executadas em bloco, sendo totalmente completas, caso a transação ocorra com sucesso, ou totalmente

desfeitas, caso ocorra um erro. De forma a identificar as operações realizadas com êxito e as que não foram finalizadas, o sensor monitora as operações de início de transação (*Begin Transaction*), se a mesma foi concluída com êxito (*Commit*), ou se foi abortada (*Rollback*).

- **Modificação**: especifica as operações realizadas que modificam o estado dos artefatos armazenados na base de dados. Essas operações podem ser **criação** (indica que um novo artefato foi criado na aplicação cooperativa, exigindo que novos itens sejam inseridos nos artefatos da base de dados); **atualização** (indica que um artefato da aplicação cooperativa foi alterado, provocando conseqüentes mudanças nos itens correspondentes a esse artefato na base de dados); **remoção** (indica que um artefato da aplicação cooperativa foi removido e, por conseqüência, também o item correspondente a esse artefato na base de dados);
- **Leitura**: operações relacionadas a leitura (consulta) aos artefatos armazenados.
- **Exceção**: as operações podem ser executadas com sucesso, ou podem levantar exceções durante sua execução. Por exemplo, a aplicação pode solicitar a exclusão de um artefato que tem dependência com um outro artefato, o qual não pode ser removido, ou, a aplicação pode solicitar a alteração de um artefato que já está bloqueado por outra transação. Assim, o sensor deve monitorar, também, as **exceções** que por ventura ocorram durante a execução de cada operação.

Além do monitoramento do processo de persistência da aplicação, consideramos relevante, não para os usuários da aplicação cooperativa, mas para usuários que atuem no desenvolvimento cooperativo dessas aplicações, que operações relativas a mudanças no esquema da base de dados também sejam monitoradas. Assim, imaginamos dois tipos diferentes de sensores para Ariane:

- **Sensor de persistência**: que deverá atuar sobre o mecanismo de persistência da aplicação cooperativa, monitorando ações sobre os artefatos armazenados dessas aplicações (pertencentes aos cinco tipos descritos anteriormente).
- **Sensor de esquema**: que deverá atuar mais diretamente sobre ferramentas de definição de esquema oferecidas pelo SGBD, monitorando ações executadas pelos desenvolvedores dessas aplicações na definição da estrutura (esquema) da base de dados da aplicação.

Um requisito importante dos sensores é que eles possam ser acoplados ou desacoplados da aplicação que está sendo monitorada de forma não dependente. Isso significa que, caso a aplicação deseje remover o sensor, esta deverá poder fazê-lo sem grandes interferências em suas funcionalidades.

Uma grande facilidade da abordagem de monitoramento baseado em sensores é que toda a lógica de monitoramento fica encapsulada no próprio sensor. Isso garante uma maior reutilização desse componente e uma maior flexibilidade no monitoramento de diferentes tipos de eventos e aplicações. Sensores poderiam ser desenvolvidos para monitorar a presença e disponibilidade dos usuários do grupo, ou para monitorar outras ações sobre a aplicação (que não façam parte do processo de persistência), ou ainda para monitorar dispositivos do mundo real. Paul Saffo em (SAFFO, 2002) prevê que, em um futuro próximo, o uso de dispositivos reais dotados de sensores se tornará parte integrante dos ambiente pessoais e de trabalho.

Diversos mecanismos voltados para apoiar a percepção utilizam monitoramento baseado em sensores, como o Nessie (PRINZ, 1999) e o TelePresence (GELLERSEN e BEIGL, 1999).

#### III.5.2.2. Tratador de Eventos

O componente Tratador de Eventos é o responsável pela geração e distribuição dos eventos.

A partir do tipo de operação capturada pelo sensor, e das informações capturadas sobre essa operação, o tratador de eventos gera um evento de tipo correspondente. Assim, operações relacionadas a sessão, transação, atualização, consulta e exceção produzirão tipos correspondentes de eventos (evento de sessão, evento de transação, evento de atualização, evento de consulta e evento de exceção).

A separação desses eventos em tipos facilita o consumo dos mesmos pelos componentes de apresentação e permite que esses componentes filtrem os tipos de eventos que estão interessados.

Além de gerar os eventos, o tratador de eventos também é responsável por distribuí-los aos componentes de apresentação e ao componente tratador de armazenamento (os quais serão detalhados a seguir). Para isso, ele deve possibilitar que esses componentes se registrem a determinados tipos de eventos e sempre que eventos

desse tipo forem gerados, ele deve disparar notificações, através da rede, para os componentes registrados.

### III.5.2.3. Tratador de Armazenamento

Esse componente é responsável pela manipulação da base de dados de eventos, armazenando os novos eventos gerados pelo tratador de eventos, e permitindo que consultas possam ser realizadas sobre os eventos históricos, de modo a alimentar os componentes visuais de percepção.

Para a realização dessas consultas ele permite que alguns critérios de seleção sejam utilizados como filtros sobre esses eventos. Por exemplo, o componente pode selecionar os eventos, baseado em um dos seguintes critérios, ou um conjunto deles:

- **Tempo**: eventos que tenham ocorrido em um determinado intervalo de tempo;
- **Artefato**: eventos que representam ações sobre um determinado artefato;
- **Usuário**: eventos produzidos por um usuário específico;
- **Tipo**: eventos que indiquem uma determinada operação.

### III.5.2.4. Componentes visuais de percepção

Para que a percepção ocorra, os eventos ocorridos devem ser consumidos, ou seja, apresentados aos usuários finais da aplicação cooperativa. Os componentes visuais têm sido amplamente utilizados para apresentação das informações de percepção em aplicações cooperativas, conforme discutido na seção **II.3.8.1**. Seu sucesso deve-se ao fato deles serem pequenos, leves e focados em funcionalidades específicas.

Como o propósito de Ariane não é oferecer soluções específicas para a questão de apresentação das informações de percepção, a idéia é utilizar diferentes componentes visuais de percepção, propostos e conhecidos, alimentando-os com os eventos produzidos pelo servidor de percepção. Para poder aceitar a notificação desses eventos de modo síncrono (no instante em que os eventos sejam produzidos), esses componentes deverão se registrar junto ao servidor de percepção, indicando os tipos de eventos desejados. Para apresentar a informação produzida de forma assíncrona (em um momento posterior à geração dos eventos), os componentes visuais podem realizar consultas ao histórico de eventos, através do tratador de armazenamento.

Como os eventos são representados seguindo uma estruturação padrão, os componentes visuais terão facilidade em tratar e interpretar as ações que originaram os eventos, de acordo com o contexto específico da aplicação cooperativa ao qual estão associados.

### III.5.2.5. BD de Eventos - Base de dados histórica dos eventos

O BD de Eventos é um repositório de dados onde são persistidos todos os eventos produzidos ao longo das interações do grupo. Sohlenkamp, em (SOHLENKAMP, 1998), apresenta cinco vantagens em manter o histórico dos eventos ocorridos em um sistema:

(a) **Indicativo de atividades passadas**: os usuários podem localizar, rapidamente, onde atividades ocorreram;

(b) **Repetição**: permite que usuários reconstruam atividades passadas próprias ou dos outros, auxiliando o entendimento do estado atual do artefato;

(c) **Prestação de contas**: O histórico pode ser usado como prova legal de que o usuário foi responsável por uma determinada ação;

(d) **Entendimento**: Usuários podem não estar disponíveis todo o tempo. Assim, os usuários que não tenham participado de uma interação podem procurar pelo que aconteceu durante sua ausência;

(e) **Resolução de conflitos de versões**: mudanças individuais podem ser mantidas para decidir, em caso de conflito, quais das ações conflitantes devem ser mantidas e deixar que o sistema ofereça algum apoio para a construção de novas versões.

### III.5.2.6. Cubo de Percepção – Histórico multidimensional dos eventos

No decorrer das diversas interações dos usuários sobre a aplicação cooperativa, um volume bastante grande de eventos é gerado. Após a coleta desses dados é possível extrair informações escondidas ou valor agregado. Entretanto, sem ferramentas de apoio é quase impossível analisar tamanha quantidade de informação. Dentre as ferramentas de análise existentes, os sistemas de processamento analítico (OLAP – *Online Analytical Processing*) têm se destacado, por sua capacidade de realizar consultas complexas, cálculos, comparações e agregações sobre várias dimensões dos dados

(CAMPOS, 2000; ÖZSU e VALDURIEZ, 1999; RAMAKRISHNAN e GEHRKE, 2000).

Ariane oferece uma estrutura para armazenamento das informações de percepção, produzidas e armazenadas no BD de Eventos, que viabiliza a análise dessas informações através de sistemas OLAP. Essa estrutura é o Cubo de Percepção. A estrutura de cubos, onde os dados são modelados de forma multidimensional, é comumente utilizada em sistemas OLAP, devido à possibilidade de analisar os dados sob várias dimensões.

O Anexo 2 apresenta alguns conceitos e termos relacionados à modelagem multidimensional e aos sistemas OLAP. Informações adicionais podem ser conseguidas junto ao grupo DataWare (DATAWARE, 2002).

#### III.5.2.7. Processador ETL

Para que as informações de percepção contidas no BD de Eventos sejam transpostas para o formato multidimensional e armazenadas no Cubo de Percepção, elas devem passar por um processo, conhecido como ETL – Extração, Transformação e Carga. O processador ETL é o elemento da arquitetura responsável pela extração dos eventos do BD de Eventos, pela sua transformação para o formato multidimensional e sua carga no Cubo de Percepção. Esse processo deve ser iniciado, ou ter seu início programado, pelo usuário administrador dos dados da aplicação.

## III.6. ANÁLISE DO MECANISMO DE PERCEPÇÃO ARIANE

O mecanismo de percepção Ariane é caracterizado como uma infra-estrutura genérica de percepção. A seguir é apresentada uma comparação de Ariane, em relação a três outras infra-estruturas genéricas de percepção, presentes na literatura, o SISCO (MARIANI, 1997), o BW (PINHEIRO, 2001) e o Nessie (PRINZ, 1999), que foram analisados na seção II.3.8.2.

Similar ao SISCO (MARIANI, 1997), Ariane propõe o monitoramento de ações sobre bases de dados. No entanto, ao contrário deste, não está limitado a um SGBD particular, nem ao modelo de dados orientado a objetos.

Ariane difere do BW (PINHEIRO et al., 2002) por que este é proposto como um *framework* de percepção, o qual requer que extensões sejam feitas para permitir sua

utilização por uma aplicação cooperativa, enquanto que o propósito de Ariane é ser um mecanismo "caixa preta", ao qual a aplicação cooperativa pode se conectar ou desconectar para utilizar seus serviços de percepção. Caso necessário, o mecanismo também poderá ser estendido, para atender demandas especiais de uma determinada aplicação. Além disso, ao contrário do BW, onde a aplicação deve informar a ocorrência dos eventos explicitamente, Ariane utiliza a abordagem de coleta implícita para monitoramento das ações, onde a coleta é feita de forma não intrusiva, transparente para a aplicação, como parte do seu processo de persistência.

Em relação ao Nessie (PRINZ, 1999), uma idéia importada dessa abordagem é o uso de sensores para monitoramento das ações tratadas no mecanismo de percepção. Entretanto, a implementação desses sensores, em Ariane, segue um paradigma totalmente diferente do utilizado em Nessie. No Nessie, os sensores devem ser programados pela aplicação cooperativa e os eventos são provenientes de um cliente específico do Nessie, ou devem ser comunicados diretamente pela aplicação ao servidor de percepção, através de HTTP e *scripts* CGI. Enquanto que Ariane utiliza a abordagem de coleta implícita para obtenção das informações de percepção.

# *Capítulo IV*

## *O Protótipo de Ariane*

*No capítulo anterior, foi apresentado o mecanismo de percepção Ariane. Neste capítulo, é detalhado como um protótipo de Ariane foi implementado e como ele foi utilizado em um ambiente de desenvolvimento colaborativo de componentes, o OdysseyShare. São discutidas as decisões de projeto tomadas durante o desenvolvimento e as tecnologias empregadas para alcançar os requisitos propostos.*

### IV.1. INTRODUÇÃO

Esta dissertação está inserida em um contexto maior, o projeto *OdysseyShare*. Por essa razão, alguns requisitos técnicos direcionaram a implementação do protótipo de Ariane. Um desses requisitos foi que, para facilitar a integração com o ambiente *OdysseyShare*, a tecnologia utilizada fosse compatível com a plataforma Java. Além disso, um dos princípios adotados no desenvolvimento desse protótipo foi o de reutilização de *software*. Para isso, foi despendido um tempo maior em pesquisa e estudo de soluções tecnológicas inovadoras que simplificassem o desenvolvimento e ampliassem os serviços do protótipo. Essas tecnologias são descritas a seguir.

Por questões de estilo e de coerência entre a linguagem de programação e as variáveis definidas, foi utilizado o inglês como idioma base para a codificação do protótipo. Assim, os diagramas de classe apresentados aparecem nesse idioma.

Para apresentar como o protótipo de Ariane foi implementado, este capítulo está organizado da seguinte maneira: a seção IV.2 discute como foi definido e implementado o monitoramento das ações da aplicação cooperativa sobre o SGBD; a seção IV.3 mostra como foram implementados os componentes do servidor de percepção; a seção IV.4 apresenta a utilização de Ariane junto ao ambiente cooperativo *OdysseyShare* e a seção IV.5 conclui o capítulo discutindo algumas utilizações potenciais de Ariane.

## IV.2. MONITORAMENTO

Um ponto central na implementação de Ariane é o Sensor, que é o módulo responsável pelo monitoramento e coleta das ações da aplicação sobre o SGBD. Para garantir que o monitoramento ocorra de modo não intrusivo e transparente tanto para a aplicação quanto para o SGBD, dois problemas tiveram que ser analisados para a implementação do sensor: (1) Como monitorar o que ocorre no SGBD sem criar dependência a um SGBD específico, ou seja, em que ponto da comunicação entre a aplicação e o SGBD o sensor deveria ser acoplado? (2) Como implementar o sensor de forma a garantir o monitoramento transparente e a coleta implícita? As duas próximas seções discutem cada uma dessas questões.

### IV.2.1. Onde Acoplar o Sensor?

A Figura 16 apresenta uma visão geral da arquitetura de um SGBD. Os usuários finais podem manipular os dados armazenados em um banco de dados físico através de: (i) funções pré-definidas em uma aplicação de negócio (cooperativa ou não); ou (ii) aplicações utilitárias fornecidas pelo próprio SGBD. Estamos interessados, apenas, em ações executadas pelos usuários através de aplicações de negócio. Uma interface de programação (API - *Application Programing Interface*), compatível com o SGBD, fornece a ponte de comunicação entre essas aplicações e o SGBD propriamente dito.

Mecanismos de persistência transparente são sistemas que abstraem os serviços de persistência para fora das aplicações de negócio, possibilitando que essas aplicações trabalhem com objetos persistentes diretamente da mesma forma que o fazem com outros objetos da aplicação (TYAGI et al., 2003). Os detalhes sobre como esses objetos são persistidos em um SGBD são implementados pelo mecanismo de persistência, ficando transparente para a aplicação a maneira como isso é feito.

Para que aplicações Java se comuniquem com um SGBD, elas devem utilizar a API JDBC (*Java Database Connectivity*) (SUN, 2003d). Essa API implementa uma série de classes e interfaces que provê uma interface padrão para desenvolvedores de SGBDs e de aplicações. Com isso, uma mesma aplicação Java pode funcionar acessando diferentes SGBDs.

Apresentado esse cenário de persistência, percebe-se que existem, basicamente, quatro pontos na comunicação entre uma aplicação e um SGBD, onde o monitoramento

das ações executadas pela aplicação sobre o SGBD pode ser realizado: (a) na própria aplicação do usuário; (b) no mecanismo de persistência transparente (supondo que a aplicação utilize um); (c) na interface JDBC (aplicações Java); ou (d) no SGBD propriamente dito (monitorando sua API de comunicação ou utilizando serviços oferecidos pelo SGBD). Cada uma dessas opções de monitoramento será discutida nas próximas seções.

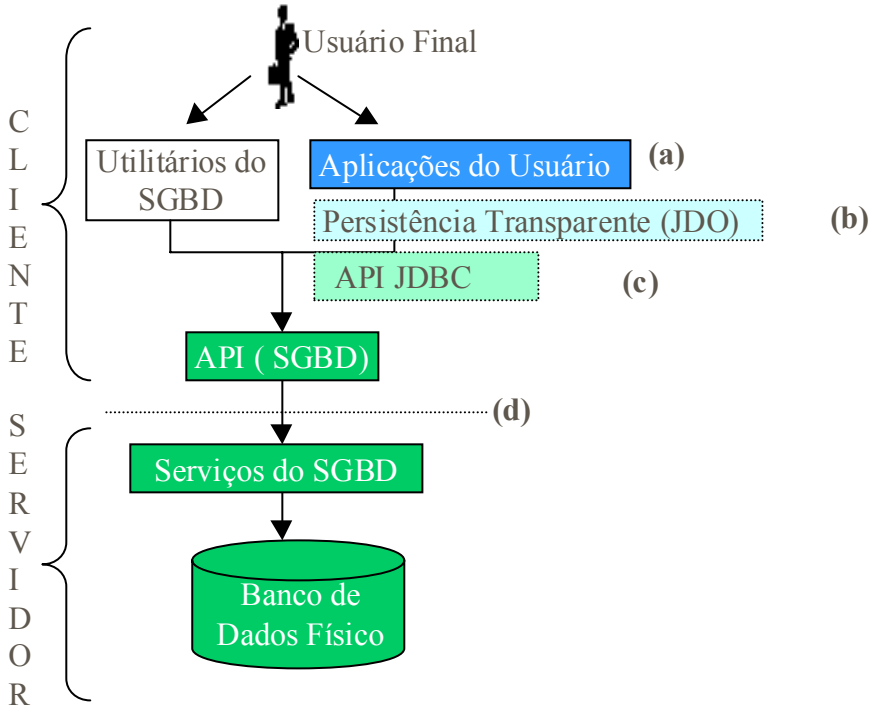



**Figura 16 – Visão geral da arquitetura de um SGBD**

### IV.2.1.1. Aplicação do Usuário

Essa é a forma de monitoramento adotada pela maioria dos mecanismos de percepção. Nesse caso, a aplicação é quem indica, em seu código, que funções realizam a execução de ações sobre o SGBD e monitora a execução dessas funções, recuperando as informações que indiquem a ocorrência de cada operação sobre o SGBD. O sensor teria que ser construído, então, como uma extensão à aplicação, de forma fortemente acoplada à mesma.

A principal vantagem em monitorar diretamente a aplicação é que as informações coletadas trazem em si a semântica por trás das ações, facilitando o consumo posterior da informação de percepção. Por exemplo, a ação "remover a classe 'Pessoa' do

diagrama ‘X’”, executada pela aplicação, pode desencadear uma série de comandos *delete* no banco de dados. Se o monitoramento é feito na aplicação, a informação obtida é o que está descrito na ação (que do ponto de vista da aplicação é o que o usuário efetivamente precisa saber). Se o monitoramento for sobre o conjunto de comandos *delete*, torna-se necessária uma etapa de tradução desses comandos para produzir a informação original sobre a ação.

Porém, essa opção tem a desvantagem de limitar o uso de Ariane ao contexto de uma aplicação específica, o que foge ao primeiro requisito estabelecido (ser independente de aplicação e SGBD).

#### IV.2.1.2. Mecanismo de Persistência Transparente

Esses mecanismos atuam sobre o modelo de objetos da aplicação, realizando o mapeamento para o modelo de persistência e vice-versa. Toda a lógica de persistência fica implementada nesses sistemas.

*Java Data Objects* (JDO) (RUSSELL, 2002) é uma especificação de um conjunto de interfaces destinadas a padronizar a implementação de mecanismos de persistência transparente em aplicações Java. Diversas implementações de JDO vêm sendo desenvolvidas, algumas de código aberto, como o TJDO (TJDO, 2003) e o OJB (APACHE, 2003), e outras comerciais, como o JDO *Genie* (HEMISPHERE, 2003), o Kodo (SOLARMETRIC, 2003) e o LiDO (LIBELIS, 2003). Essas implementações abrangem os principais SGBDs relacionais. Alguns SGBDs orientados a objetos como o ObjectDB (OBJECTDB, 2003), o Versant (VERSANT, 2003) e o Gerente de Objetos Armazenados GOA (MATTOSO et al., 2002), também vêm adotando uma interface de comunicação com o JDO. O surgimento de novas implementações JDO e experiências relatadas no portal *JDOCentral* (JDO CENTRAL, 2003) e em grupos de discussão (JDO COMMUNITY, 2003), atestam que o modelo de persistência proposto é válido e viável.

Dessa maneira, o sensor de Ariane poderia monitorar mecanismos de persistência transparente, em conformidade com a especificação JDO, o que garantiria a Ariane independência de aplicação e SGBD, limitando, apenas, que a aplicação e o SGBD sejam reconhecidos por alguma implementação JDO. Além disso, uma grande vantagem em realizar o monitoramento no âmbito do JDO que ele pode ser feito sobre o modelo de dados da aplicação.

#### IV.2.1.3. Interface JDBC

Uma outra forma de capturar as ações sobre o SGBD é interceptando as sentenças enviadas pela aplicação para o SGBD através da interface JDBC. Desse modo, garante-se que o monitoramento ocorra de forma independente tanto da aplicação quanto do SGBD utilizado.

Um dispositivo que efetua esse monitoramento é o P6Spy (P6SPY TEAM, 2003). O P6Spy funciona como uma ponte entre a aplicação de negócio e o *driver* JDBC do SGBD utilizado pela aplicação (sendo, ele mesmo, um *driver* JDBC). Todos os comandos enviados pela aplicação para o SGBD, e as respostas deste para a aplicação, passam, obrigatoriamente, pelo P6Spy, o qual pode tratá-los da maneira desejada. Assim, o sensor de Ariane poderia estender o P6Spy (ou um dispositivo similar), interceptando os comandos JDBC.

Essa opção garante a independência do mecanismo de percepção em relação à aplicação e ao SGBD, além de permitir capturar a maioria dos eventos previstos para Ariane (sessão, transação, mudanças e consulta dos dados). Eventos de mudanças no esquema do banco de dados da aplicação não podem ser capturados, a menos que a aplicação que gerencie o esquema também suporte JDBC.

A principal desvantagem dessa solução é que os comandos tratados pelo JDBC, geralmente, estão escritos na linguagem SQL. Para obter as informações que permitam identificar a ação ocorrida, é necessário criar um mecanismo que traduza os comandos SQL. Além disso, esses comandos são construídos considerando o modelo de persistência e não o modelo de objetos da aplicação, obrigando que um mapeamento para esse modelo também seja feito.

#### IV.2.1.4. SGBD

O SGBD possui conhecimento sobre todas as ações realizadas nas bases de dados que gerencia, e, internamente, já realiza o monitoramento dessas ações gravando-as em arquivos de *log*. Uma forma de utilizar as informações monitoradas pelo SGBD em Ariane seria estendendo o SGBD (ou apenas a API fornecida pelo SGBD) com o sensor. Com isso, o sensor poderia capturar as solicitações à medida que chegassem ao SGBD enviando as informações coletadas para o servidor de percepção.

Essa opção é interessante do ponto de vista de independência da aplicação, pois as operações executadas por qualquer aplicação sobre o SGBD seriam monitoradas, além de possibilitar que todos os eventos, incluindo os relativos a mudança de esquema, sejam capturados.

Por outro lado, o mecanismo de percepção ficaria totalmente dependente de um SGBD particular, o que também foge aos requisitos de Ariane. Além disso, as operações capturadas no âmbito do SGBD referem-se ao modelo de persistência da aplicação, sendo necessário um passo adicional de mapeamento do modelo de persistência para o modelo de objetos da aplicação para que a informação de percepção seja interpretada e apresentada aos usuários.

***Gatilhos (Triggers)***

A maioria dos SGBDs oferece um mecanismo que possibilita que regras sejam programadas nos artefatos do SGBD as quais iniciam alguma ação quando uma dada condição é satisfeita (STAAB e BARNEKOW, 1999). Esse mecanismo é o gatilho (*trigger*). Gatillhos, em geral, são disparados quando operações de modificação são executadas sobre artefatos de um SGBD, executando a ação programada para o artefato modificado. Assim, ao invés de estender a API do SGBD (o que nem sempre é viável) uma outra opção seria que o sensor fosse programado utilizando gatilhos.

O uso de gatilhos para o monitoramento permite uma independência da aplicação monitorada (pois a programação ocorre no âmbito do SGBD) e é de fácil implementação (não exigindo que mudanças internas sejam realizadas no SGBD).

No entanto, gatilhos apresentam diversos problemas para apoiar o mecanismo Ariane. O principal deles é a alta dependência do SGBD (gatilhos implementados em um SGBD, em geral, não podem ser transpostos para outro) e a necessidade de implementar um gatilho para cada operação (criação, atualização, remoção) sobre cada artefato armazenado no SGBD. Assim, se o banco de dados da aplicação possui 150 tabelas é necessário criar 450 gatilhos (três para cada tabela). Se houver mudanças no esquema da tabela os gatilhos correspondentes terão que ser recriados. Além disso, apenas eventos de modificação nos dados dos artefatos podem ser monitorados.

### IV.2.1.5. Discussão sobre as opções de monitoramento

Todas as opções possuem vantagens e desvantagens (Tabela 1). Para o protótipo de Ariane, optou-se por utilizar monitoramento sobre mecanismos de persistência transparente compatíveis com a especificação JDO.

Essa escolha deve-se a essa solução poder ser reutilizada por diversas aplicações acessando diferentes SGBDs, além de acreditar-se que o JDO se tornará um padrão de fato para persistência transparente em aplicações Java (linguagem que vem se firmando como um padrão para o desenvolvimento de aplicações). A decisão de usar o JDO é fundamental para que o monitoramento ocorra sobre o modelo de dados da aplicação, ao invés de ocorrer sobre o modelo de dados da persistência. Com isso, as informações de percepção capturadas são de mais fácil interpretação pelo usuário da aplicação, já que a semântica é a mesma.

**Tabela 1 - Quadro comparativo das opções de monitoramento**

| | (a) | (b) | (c) | (d) | |
|---|---|---|---|---|---|
| **Requisitos de Ariane** | **Aplicação** | **JDO** | **JDBC** | **SGBD** | |
| | | | | **API** | **Gatilho** |
| Independência da aplicação | ✗ | ✓ | ✓ | ✓ | ✓ |
| Independência do SGBD | ✗ | ✓ | ✓ | ✗ | ✗ |
| Modelo de dados da aplicação | ✓ | ✓ | ✗ | ✗ | ✗ |
| Eventos de Sessão e Transação | ✓ | ✓ | ✓ | ✓ | ✗ |
| Eventos CRUD | ✓ | ✓ | ✓ | ✓ | ✓ |

## IV.2.2. Implementação do Sensor

Uma vez escolhido realizar o monitoramento sobre mecanismos de persistência transparentes compatíveis com JDO, surge a questão de como implementar o sensor de maneira que esse monitoramento seja realizado no âmbito da especificação JDO.

Para capturar as ações do SGBD sobre uma implementação JDO específica, teríamos que ter acesso ao código fonte da mesma e entender como a especificação JDO foi implementada, identificando os pontos da implementação onde o código de monitoramento deveria ser incluído. Aqui temos alguns problemas: (i) dificuldade de acesso ao código-fonte, uma vez que as implementações mais robustas são comerciais e

não disponibilizam seu código; (ii) forte dependência da implementação; (iii) evoluções na especificação ou na implementação poderiam impactar no funcionamento do serviço ou demandariam alto esforço de manutenção do mecanismo.

Nota-se que uma solução mais eficiente deveria ser imaginada. A partir de várias discussões sobre esse problema e de sugestões obtidas em fórum de discussão sobre JDO (JDO COMMUNITY, 2003) surgiu a idéia de implementar os sensores como aspectos. Aspectos fazem parte de um novo paradigma de desenvolvimento de aplicações chamado Programação Orientada a Aspectos (AOP – *Aspect Oriented Programming*) (KICZALES et al., 1997).

As próximas seções apresentarão alguns conceitos que regem a AOP e mostrarão como foi realizada a implementação do sensor, como aspecto, para monitorar operações realizadas por qualquer implementação JDO.

#### IV.2.2.1. Programação Orientada a Aspectos

A AOP representa um novo paradigma no desenvolvimento de aplicações e tem por objetivo "apoiar o programador em componentes e aspectos claramente separados um do outro, através de mecanismos que possibilitem abstraí-los e compô-los de forma a produzir o sistema como um todo" (KICZALES et al., 1997). Com isso, o desenvolvedor tem condições de, claramente, separar componentes (objetos) de aspectos (conceitos) (GARCIA et al., 2002).

A AOP foi desenvolvida para que os problemas gerados por ter que implementar várias vezes o mesmo trecho de código, em diversos pontos de uma aplicação, fossem reduzidos através da introdução de uma nova unidade de programação, denominada **aspecto**. Aspectos são características que atravessam as funcionalidades básicas de um sistema, ou seja, eles não são características de apenas um componente, mas estão presentes em vários.

A premissa da AOP é possibilitar o encapsulamento dos requisitos transversais (*crosscutting concerns*) de uma aplicação, ou seja, os requisitos que ficam espalhados em diversos pontos na hierarquia de classes da aplicação. Exemplos de requisitos transversais típicos de uma aplicação são: depuração, *logging*, persistência, controle de acesso, concorrência, etc.

Em Ariane, o componente Sensor necessita incluir o código de monitoramento em diversos trechos de código de uma implementação JDO (os pontos onde esse código será adicionado são obtidos a partir das definições na especificação JDO), que indicam que ações foram executadas sobre um SGBD. O código para realizar esse monitoramento estará espalhado por diversas classes dentro da implementação JDO. Assim, conceitualmente, é bastante intuitivo imaginar os sensores como aspectos.

O Anexo 1 descreve os principais conceitos relacionados à AOP e à ferramenta que implementa AOP para Java, utilizada no protótipo, AspectJ.

### IV.2.2.2. Implementação de Sensores como Aspectos

A exemplo da Programação Orientada a Objetos (POO), a AOP permite o uso de herança. A Figura 17 mostra a hierarquia dos sensores criados, segundo a representação de aspectos em UML sugerida em (SUZUKI e YAMAMOTO, 1999). Ainda não há uma representação consensual UML para o projeto de aspectos, mas diversas extensões à UML estão sendo propostas na literatura (CHAVEZ e LUCENA, 2001; HO et al., 2000).

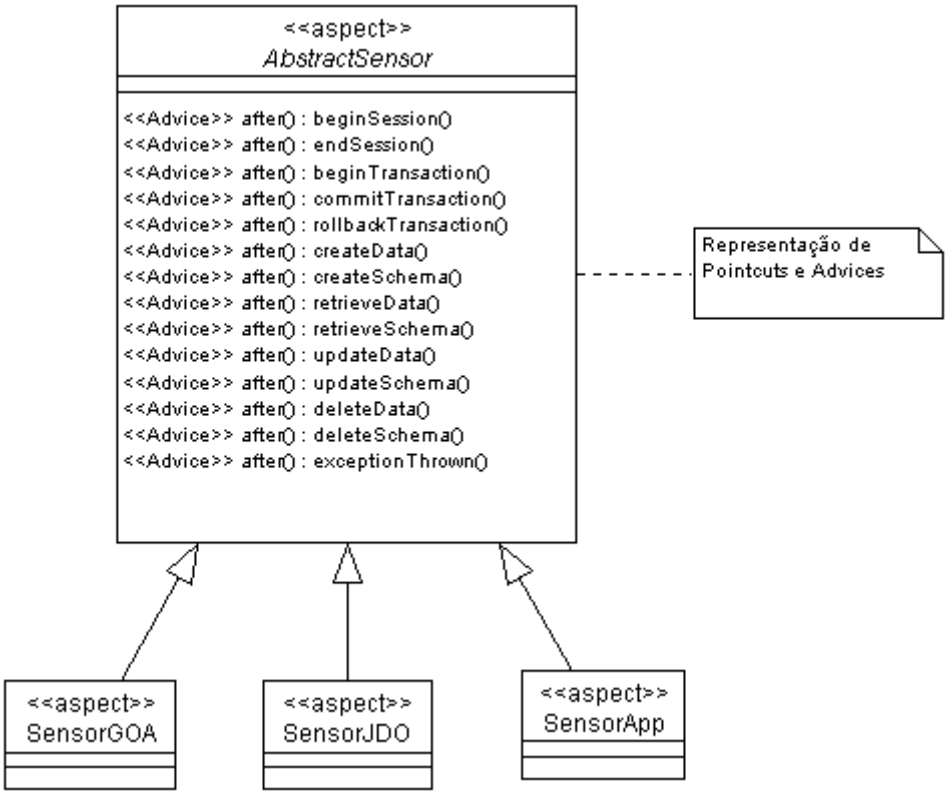



**Figura 17 – Diagrama de Classes do Sensor**

Foi criado um sensor abstrato que define o conjunto de operações especificado para Ariane e implementa a comunicação das ações capturadas para o servidor de percepção. O uso de herança e da AOP facilita que diferentes sensores filhos sejam construídos com relativa simplicidade. Basta que esses sensores indiquem os pontos de acesso na aplicação específica que desejem monitorar e programem a extração das informações sobre as ações.

No protótipo de Ariane foi implementado o aspecto SensorJDO. Esse aspecto identifica, para cada tipo de evento monitorado por Ariane, os pontos de acesso correspondentes (segundo definições contidas na especificação JDO). Através da técnica de entrelaçamento (*weaving*) definida na AOP, esse aspecto pode ser acoplado a qualquer implementação JDO, mesmo que o código fonte da implementação não esteja disponível. O SensorJDO foi testado junto a duas implementações JDO, uma comercial, o JDO Genie (HEMISPHERE, 2003) e outra de código aberto, o TJDO (TJDO, 2003). Nenhuma alteração específica no SensorJDO foi necessária para se adaptar às implementações.

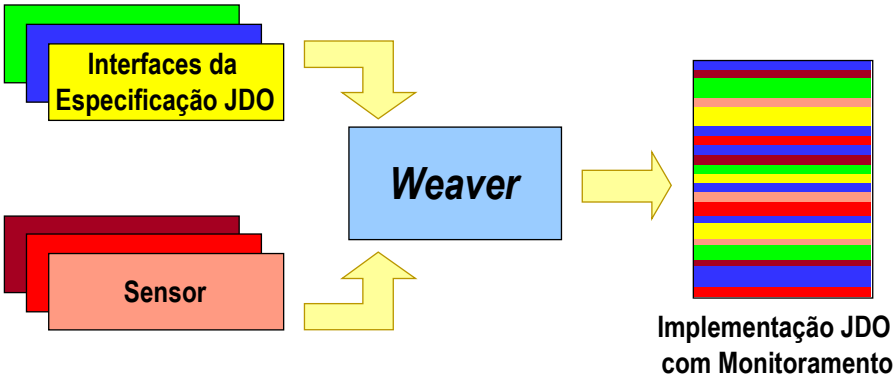



**Figura 18 – Ilustração do funcionamento do Sensor JDO como Aspecto**

A Figura 18 ilustra como ocorre a implementação do SensorJDO. De um lado temos uma implementação JDO qualquer, cujas classes concretizam interfaces definidas na especificação JDO. De outro lado, temos o Sensor, implementado como um aspecto, totalmente isolado e independente da implementação JDO. O Sensor define, internamente, os pontos de acesso da implementação (definidos na especificação JDO), que correspondem às operações que podem ser executadas sobre o SGBD, e nas quais Ariane tem interesse. Um pós-compilador (*Weaver*), fornecido pela ferramenta que implementa o AOP (no nosso caso, o AspectJ), faz o entrelaçamento do código do

Sensor no código das classes da implementação JDO, produzindo uma nova implementação JDO, estendida com o serviço de monitoramento.

A implementação dos sensores como aspectos mostrou-se essencial para o desenvolvimento do protótipo, e foi realizada de forma simples e intuitiva. Sem o uso da AOP seria praticamente inviável realizar o monitoramento no âmbito da especificação JDO, sem depender de acesso aos códigos fontes de uma implementação específica. Algumas vantagens observadas:

- **Portabilidade**: foi possível definir os pontos de monitoramento em função da especificação JDO, sem qualquer modificação explícita em uma implementação JDO específica. Com isso, um mesmo sensor pôde ser conectado a diferentes implementações;
- **Extensibilidade**: novos sensores podem ser implementados, com relativa simplicidade, para monitorar outras aplicações ou mecanismos de persistência;
- **Flexibilidade**: o acoplamento do sensor à implementação JDO é realizado através de um processo de pós-compilação das classes da implementação. Evoluções na implementação não afetam o funcionamento do sensor (pois ela deve se manter compatível com a especificação), e basta que seja executado novamente a pós-compilação sobre as novas versões das classes;
- **Acoplamento flexível**: o sensor é construído separadamente da implementação, podendo ser conectado ou desconectado à mesma.

No entanto, o uso da AOP apresentou, também, alguns problemas. Dentre eles:

- **Grau de maturidade da tecnologia**: por ser uma tecnologia muito recente, o AspectJ (implementação de AOP para Java utilizada) ainda não atingiu um grau de maturidade e estabilidade satisfatórios;
- **Domínio da tecnologia**: por representar um novo paradigma de desenvolvimento, houve uma certa dificuldade inicial em fazer o que se desejava dentro do que a tecnologia se propunha.

## IV.3. COMPONENTES DO SERVIDOR DE PERCEPÇÃO

Para realizar a comunicação entre os componentes distribuídos na arquitetura de Ariane (servidor de percepção, sensores e componentes visuais) foi utilizada a invocação remota de métodos (RMI)[1] (SUN, 2003b). O servidor de percepção é implementado como um objeto remoto e, ao ser iniciado, a instância desse servidor é ligada a um nome no *RMIRegistry*[2] para que possa ser localizado.

Os componentes visuais de percepção também devem ser implementados como objetos remotos e devem ser programados para aceitar chamadas de entrada (*incoming calls*), ou seja, permitir que o objeto seja localizado e contatado na rede pelo servidor de percepção para ser notificado sobre os eventos ocorridos.

A Figura 19 ilustra o diagrama de classes, segundo notação UML, do projeto de implementação do mecanismo de percepção Ariane.

### IV.3.1. Tratador de Eventos

O tratador de eventos é o módulo responsável por gerar eventos a partir das informações coletadas pelos sensores, por registrar componentes interessados nesses eventos e por notificar esses componentes quando novos eventos forem gerados.

Os tipos de eventos definidos em Ariane (Figura 20) estão associados às operações que podem ser realizadas por uma aplicação sobre um SGBD, para manipular seus artefatos armazenados. Essas operações abrangem tudo o que possa ocorrer em uma sessão de trabalho do usuário junto ao SGBD, desde sua conexão a uma base de dados, transações iniciadas e finalizadas com sucesso ou erro, operações de leitura e escrita executadas em cada transação, bem como eventuais erros na execução das operações, até a desconexão do usuário.

O modelo de notificação de eventos adotado em Ariane (Figura 21) é baseado no modelo utilizado pela plataforma Java 2 SDK para tratar eventos no *JavaBeans* (SUN, 2003c) e no *Java Abstract Windowing Toolkit* (AWT) (SUN, 2003a).

[1] RMI – *Remote Method Invocation* – é um mecanismo que permite invocar métodos de um objeto remoto, usando a mesma sintaxe se o mesmo estivesse localmente. Objetos remotos devem ser descritos por uma ou mais interfaces remotas (interfaces que herdam de *java.rmi.Remote*).
[2] *RMIRegistry* é uma aplicação que registra objetos remotos utilizando as facilidades de nomes do RMI.

Neste modelo, os eventos de percepção são propagados de origens (o tratador de eventos - *AwarenessEventHandler*) para "ouvintes" (os componentes de apresentação - *AwarenessWidget*). Cada tipo distinto de notificação de evento é definido como um método Java diferente (ex. *sessionStarted(), sessionEnded()*, etc). Esses métodos são agrupados em interfaces que herdam da interface *AwarenessListener*.

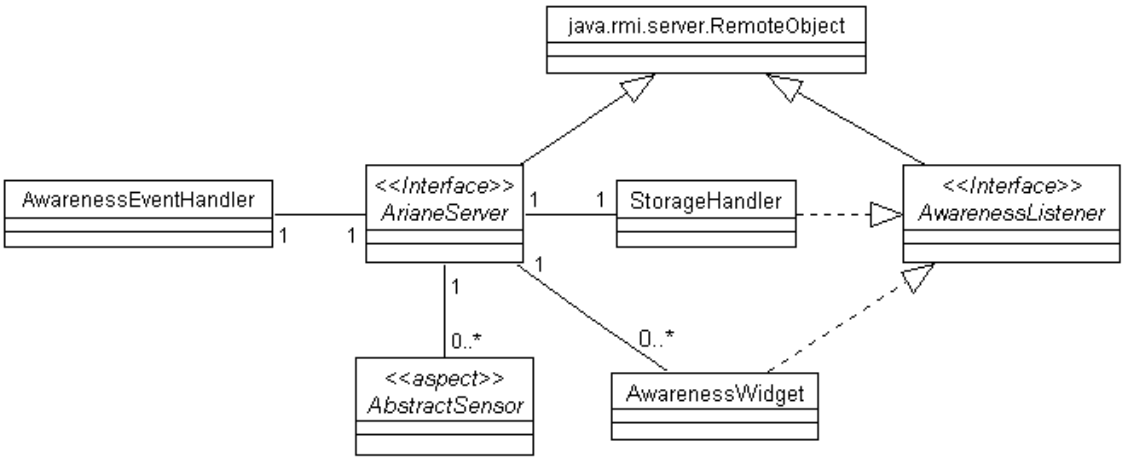



**Figura 19 – Diagrama de Classes do Mecanismo de Percepção Ariane**

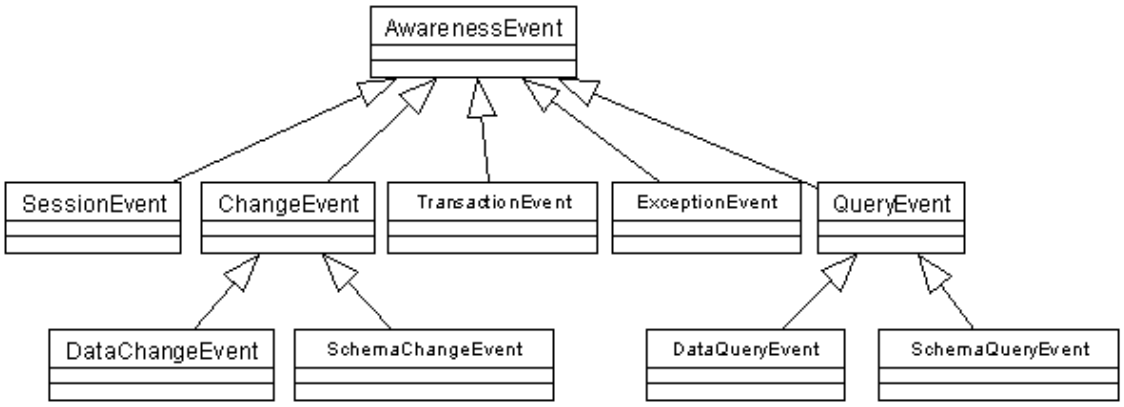



**Figura 20 – Modelo de Eventos**

Em Ariane foram definidas especializações da interface *AwarenessListener* para cada um dos tipos de eventos: *SessionListener*, *TransactionListener, DataChangeListener*, *DataQueryListener* e *ExceptionListener*. Componentes de percepção se identificam como interessados em um conjunto particular de eventos através da implementação de um conjunto dessas interfaces. Sempre que um novo evento é gerado, o tratador de eventos verifica a lista de componentes registrados para aquele tipo de evento específico e dispara uma notificação para os mesmos, através da

execução do método correspondente, especificado na interface. Esses métodos deverão ser implementados pelos componentes de percepção com o tratamento específico.

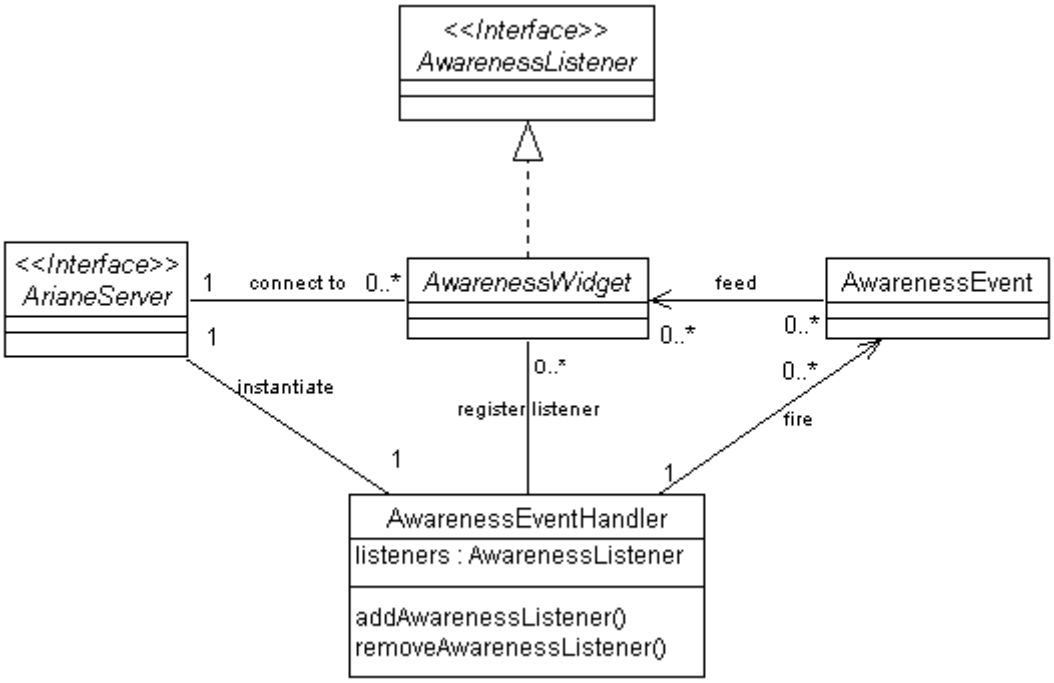



**Figura 21 – Visão Geral do Modelo de Notificação de Eventos**

Por exemplo, seja um componente de apresentação cujo objetivo é exibir o histórico de conexão dos usuários ao longo do tempo. Esse componente deve ser notificado apenas de eventos de sessão (*SessionEvent*), que indiquem que a mesma teve início ou foi encerrada. Para isso, ele deve implementar a interface *SessionListener* e seus respectivos métodos *sessionStarted()* e *sessionEnded()*. Quando for criada uma instância desse componente, a mesma deve ser registrada no tratador de eventos, através do método *addAwarenessListener*(). Feito isso, sempre que um evento de sessão ocorrer, o tratador de eventos enviará o evento às instâncias do tipo *SessionListener* registradas, através da execução dos métodos especificados na interface. No componente de apresentação, os métodos *sessionStarted()* e *sessionEnded()* devem estar programados para extrair e apresentar as informações de percepção contidas no evento.

O modo de entrega empregado nesse modelo é o *multicast*, ou seja, uma origem de eventos dispara mensagens para diversas instâncias registradas. Se ocorrer uma exceção na entrega do evento a uma instância particular, o Tratador continua a entrega aos demais componentes.

### IV.3.2. Tratador de Armazenamento

O tratador de armazenamento é implementado da mesma maneira que um componente de apresentação, ou seja, funciona como um "ouvinte" dos eventos. Ele implementa todas as interfaces do tipo *AwarenessListener* e os métodos definidos nestas interfaces.

Para controlar a persistência dos eventos, esse componente utiliza um mecanismo de persistência compatível com a especificação JDO. Para evitar chamadas recursivas ao servidor de percepção, utilizou-se uma versão da implementação JDO não estendida com os sensores, ou seja, sem o código de monitoramento. Essa possibilidade evidencia, inclusive, uma das vantagens da AOP, pois é possível utilizar duas versões da mesma implementação JDO, rodando em máquinas virtuais Java (JVM) diferentes, uma estendida com os sensores, executando no cliente, e outra sem os sensores executando no servidor de percepção.

```
<?xml version="1.0" encoding="UTF-8"?>
<jdo>
    <package name="ariane.event">
        <class name="AwarenessEvent">
            <field name="how"/>
            <field name="what"/>
            <field name="when"/>
            <field name="where"/>
            <field name="who"/>
            <field name="why"/>
        </class>
        <class name="ChangeEvent" persistence-capable-superclass="AwarenessEvent" />
        <class name="DataChangeEvent" persistence-capable-superclass="ChangeEvent" />
        <class name="DataQueryEvent" persistence-capable-superclass="QueryEvent" />
        <class name="ExceptionEvent" persistence-capable-superclass="AwarenessEvent" />
        <class name="QueryEvent" persistence-capable-superclass="AwarenessEvent" />
        <class name="SchemaChangeEvent" persistence-capable-superclass="ChangeEvent" />
        <class name="SchemaQueryEvent" persistence-capable-superclass="QueryEvent" />
        <class name="SessionEvent" persistence-capable-superclass="AwarenessEvent" />
        <class name="TransactionEvent" persistence-capable-superclass="AwarenessEvent" />
    </package>
</jdo>
```

**Figura 22 - Descrição do modelo de persistência de Ariane**

Para realizar o armazenamento dos eventos usando JDO, é necessário descrever as classes que serão persistidas em um arquivo de metadados, usando a linguagem XML, segundo um DTD proposto na especificação. A Figura 22 mostra essa definição para as classes de evento de Ariane.

Durante o desenvolvimento foi testado o armazenamento em dois SGBDs: um comercial, o Microsoft SQL Server, e outro gratuito, o PostgreSQL. Essa portabilidade é uma das vantagens da API JDO, pois a mudança de SGBD não afeta em nada a aplicação, uma vez que todo o tratamento de persistência é realizado pela API.

### IV.3.3. Geração do Cubo de Percepção

O Cubo de Percepção é um banco de dados multidimensional onde são armazenados os eventos produzidos por Ariane para fins de análise. A modelagem proposta para o cubo de percepção é baseada nas propostas do Cubo CRUD (estrutura para armazenamento de informações extraídas de *logs* gerados por SGBDs) (SULAIMAN, 2002) e no modelo multidimensional para *Data WebHouse* (armazenamento de informações extraídas de *logs* de servidores Web) proposto em (KIMBALL e MERZ, 2000).

Para que sejam armazenados em uma estrutura de cubo (formato que permite seu uso em ferramentas OLAP), os eventos devem ser remodelados. Uma forma bastante usual de modelagem multidimensional é o esquema estrela (INMON, 1997). Esse esquema consiste de uma única tabela central, chamada de tabela de fatos, geralmente contendo um grande volume de dados (CAMPOS, 2000), a qual está conectada a diversas tabelas satélites, chamadas de tabelas dimensões, que são tabelas que "qualificam" os fatos (CAMPOS, 2000).

O esquema estrela do cubo de percepção de Ariane é exibido na Figura 23 (extraída da ferramenta OLAP *Microsoft Analysis Services*). Os fatos são os eventos ocorridos, de forma quantificada. E as dimensões são as questões 5W+1H, que qualificam os eventos. Sobre essas dimensões é que os eventos são analisados. Para isso, deve-se adicionar tanta informação quanto possível a essas dimensões e deve-se estabelecer relações de hierarquias dentro delas.

Por exemplo, a questão *When* armazena a data e hora de ocorrência do evento. Para fins de análise, para facilitar as agregações e para otimizar o tempo das consultas, é interessante adicionar mais informações à dimensão *When*, correspondente. Assim, essa dimensão trata e armazena informações do tipo o **dia da semana** (segunda, terça, etc.), o número do dia, o **mês**, o **ano**, o **quarto de ano**, indica se o dia é **final de semana**, informa **mês/ano**, número da **semana no ano**. Com isso, podem ser feitas consultas do tipo: "quantos eventos ocorreram em um determinado dia, ou em um determinado mês, ou em um dado mês de um ano". Além disso, deve ser definida uma hierarquia entre os elementos dessa dimensão (a definição das hierarquias aparece do lado esquerdo superior da figura). Nesse caso temos que **ano > quarto de ano > mês > dia > hora > minuto**. As consultas utilizam essa hierarquia para efetuar as agregações aumentando

ou diminuindo a granularidade dos dados consultados a depender do nível selecionado na hierarquia (ex. eventos agrupados por dia, por mês, por ano, etc.). Essas operações no OLAP são chamadas de *drill down* (menor granularidade) e *roll up* (maior granularidade). Todos esses novos atributos podem ser utilizados, também, para filtragem dos dados, durante as consultas.

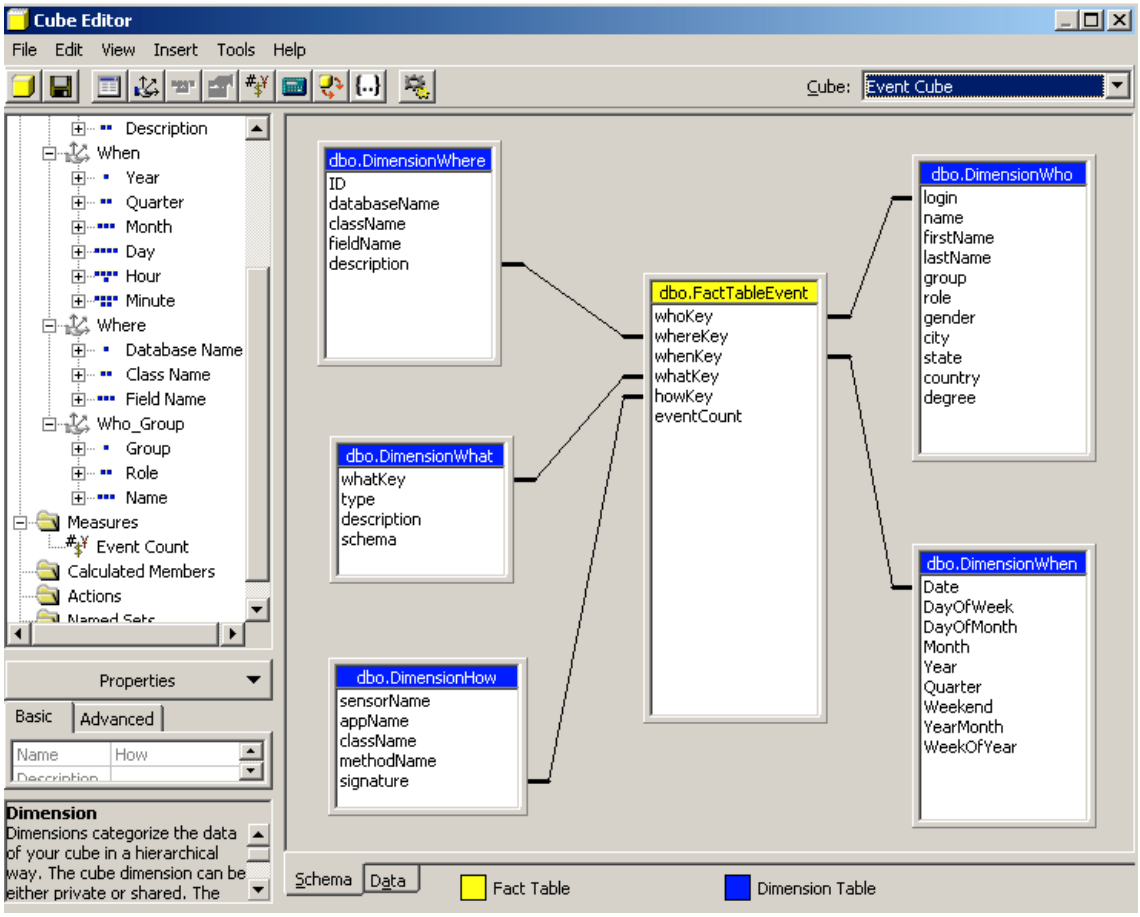



**Figura 23 – Esquema Estrela do Cubo de Percepção de Ariane**

## IV.3.4. Processador ETL

Uma vez definido o modelo multidimensional do cubo de percepção, o próximo passo é alimentar esse cubo com os dados provenientes do BD de Eventos. Para isso, é utilizado um processador ETL que irá Extrair os eventos do BD de Eventos, fará a Transformação desses eventos para o modelo multidimensional definido para o cubo, e fará a Carga dos dados transformados para cada dimensão e para a tabela de fatos.

Existem diversas ferramentas no mercado para execução dessa tarefa, como apresentado em (CAMPOS, 2000). No protótipo de Ariane optamos por utilizar as

soluções de OLAP fornecidas pelo SGBD Microsoft SQL Server. Esse SGBD disponibiliza o processador ETL Microsoft DTS (*Data Transformation Services*) (PETERSON, 2000). Esse processador permite que seja definido um pacote contendo diversas tarefas que são executadas seguindo um fluxo de processos (*workflow*) (Figura 24). Esse pacote pode ser executado manualmente, pelo administrador da aplicação cooperativa, ou por agendamento.

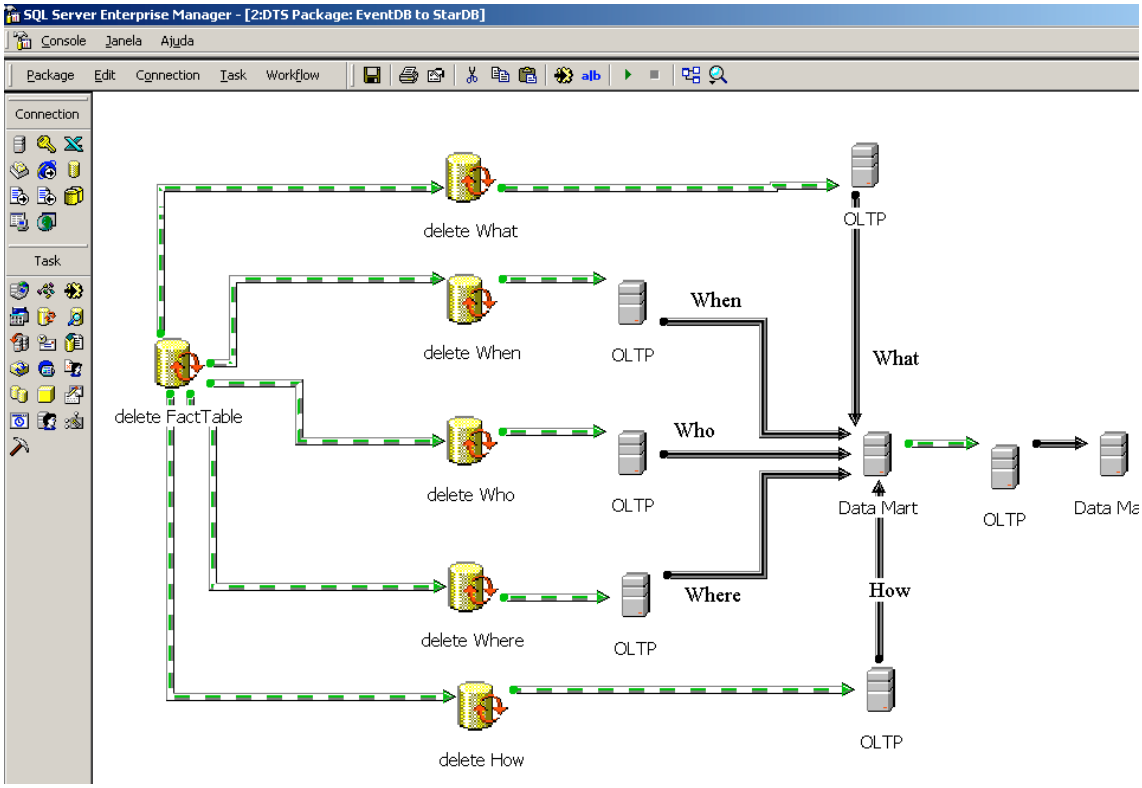



**Figura 24 – Fluxo de Processos para Geração do Cubo de Percepção Usando o DTS**

## IV.4. UTILIZAÇÃO DO PROTÓTIPO DE ARIANE NO *ODYSSEYSHARE*

Para mostrar a viabilidade e utilidade de Ariane como mecanismo de apoio à percepção, essa seção apresenta, passo a passo, um exemplo de utilização do protótipo junto ao ambiente de desenvolvimento colaborativo de componentes *OdysseyShare*. A Figura 25 mostra uma tela do ambiente *OdysseyShare*, o diagramador de casos de uso. Diversos usuários podem estar utilizando a aplicação e modelando seus diagramas. À medida que ações sejam executadas sobre artefatos persistentes essas ações são coletadas e enviadas ao servidor de percepção. Ações sobre artefatos não persistentes ou ações do tipo movimentação do cursor e de mouse, são desconsideradas. Para o usuário

o monitoramento ocorre de forma transparente, sem que o mesmo tenha que modificar seu modo de trabalho. Além disso, nenhuma modificação foi efetuada diretamente sobre a aplicação *Odyssey* para que o monitoramento pudesse ser realizado.

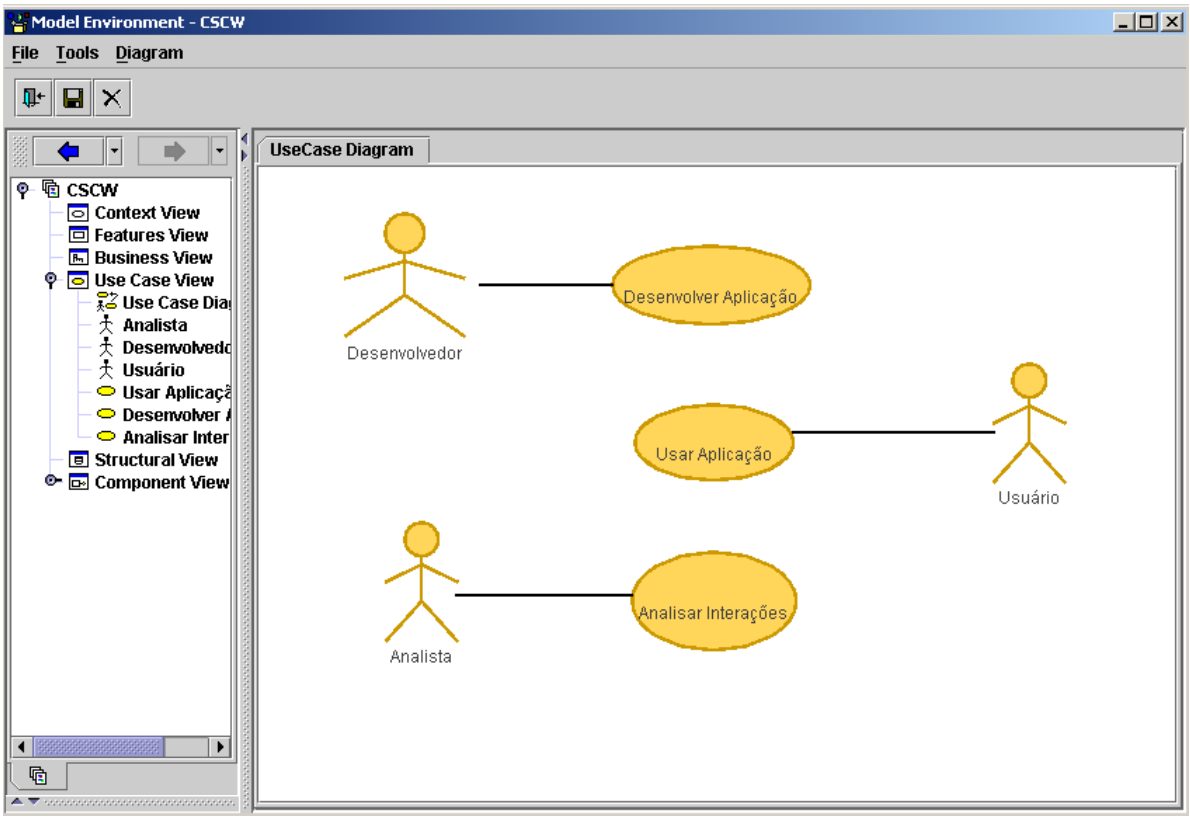



**Figura 25 – Editor de diagrama de casos de uso do *OdysseyShare***

Como o *OdysseyShare* não utilizava o JDO como mecanismo de persistência, foi necessário incluir essa opção de armazenamento no ambiente. Para isso, foi escolhida a implementação JDO Genie, por esta ter se mostrado, dentre as implementações avaliadas, a mais robusta e fácil de utilizar. Essa implementação disponibiliza um ambiente, que auxilia a identificação das classes da aplicação que devem ser persistidas. A Figura 26 ilustra a definição do modelo de persistência do *OdysseyShare* no ambiente de trabalho do JDO Genie. Na parte superior da tabela são exibidas as classes da aplicação e as opções de persistência dessa classe (como, por exemplo, a identificação da superclasse, se houver). Na parte inferior, são exibidas as configurações de persistência para cada atributo da classe. Por exemplo, deve-se indicar se o atributo é persistente; caso o atributo seja do tipo coleção, deve-se indicar qual o tipo de elemento contido nessa coleção.

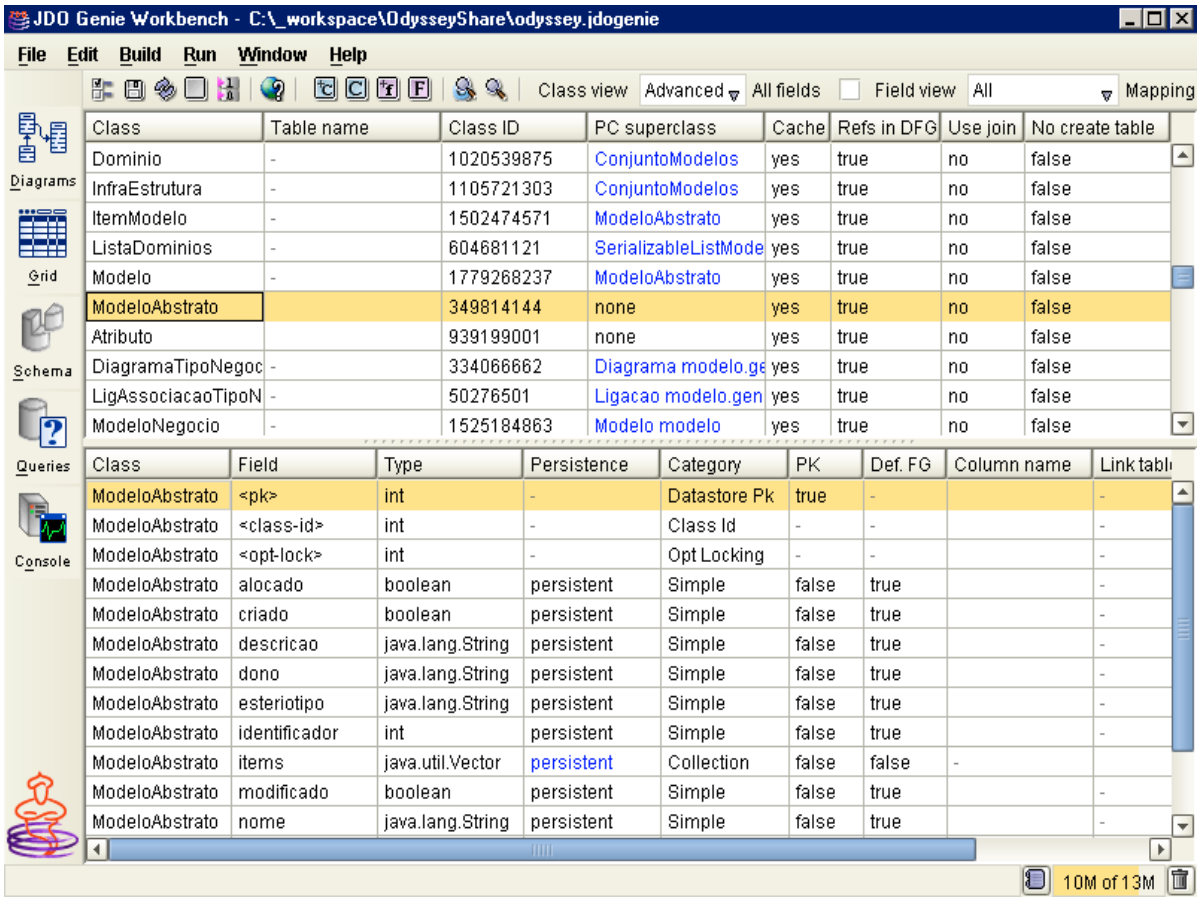



**Figura 26 – Ambiente do JDO *Genie* para o projeto de persistência do *OdysseyShare***

As configurações especificadas no ambiente do JDO Genie para as classes da aplicação servirão para gerar o arquivo de metadados (em XML), que descreve o modelo de persistência da aplicação e que é utilizado pelo JDO Genie para gerenciamento da persistência (Figura 27). Além disso, essas configurações são utilizadas para gerar, automaticamente, o esquema da base de dados (Figura 28), com as definições especificadas. O mapeamento do modelo de dados da aplicação para o modelo da persistência é feito, automaticamente, pelo ambiente do JDO Genie. Esse mapeamento aparece na Figura 28, onde é exibido ao lado do nome da tabela e de cada campo da tabela, o nome da classe e do atributo correspondentes na aplicação.

O uso do ambiente de trabalho do JDO Genie mostrou-se de fundamental importância para implementação de persistência com JDO no *OdysseyShare* pois o *OdysseyShare* é uma aplicação bastante complexa com mais de 650 classes implementadas, das quais 150, aproximadamente, foram identificadas para compor seu modelo de persistência.

```
<?xml version="1.0" encoding="UTF-8"?>
<jdo>
    <package name="modelo">
        <class name="Dominio" persistence-capable-superclass="ConjuntoModelos">
            <field name="mudancas" persistence-modifier="none" />
        </class>
        <class name="ConjuntoModelos" persistence-capable-superclass="ModeloAbstrato
        <class name="ListaDominios" persistence-capable-superclass="componentes.mode
        <class name="ModeloAbstrato">
            <field name="items" persistence-modifier="persistent">
                <collection element-type="ModeloAbstrato" />
            </field>
        </class>
        <class name="Aplicacao" persistence-capable-superclass="ConjuntoModelos" />
        <class name="InfraEstrutura" persistence-capable-superclass="ConjuntoModelos
        <class name="ItemModelo" persistence-capable-superclass="ModeloAbstrato" />
        <class name="Modelo" persistence-capable-superclass="ModeloAbstrato">
            <field name="acessados">
                <collection element-type="ModeloAbstrato" />
            </field>
            <field name="importados">
                <collection element-type="ModeloAbstrato" />
            </field>
        </class>
    </package>
```

**Figura 27 – Arquivo de descrição do modelo de persistência do *OdysseyShare***

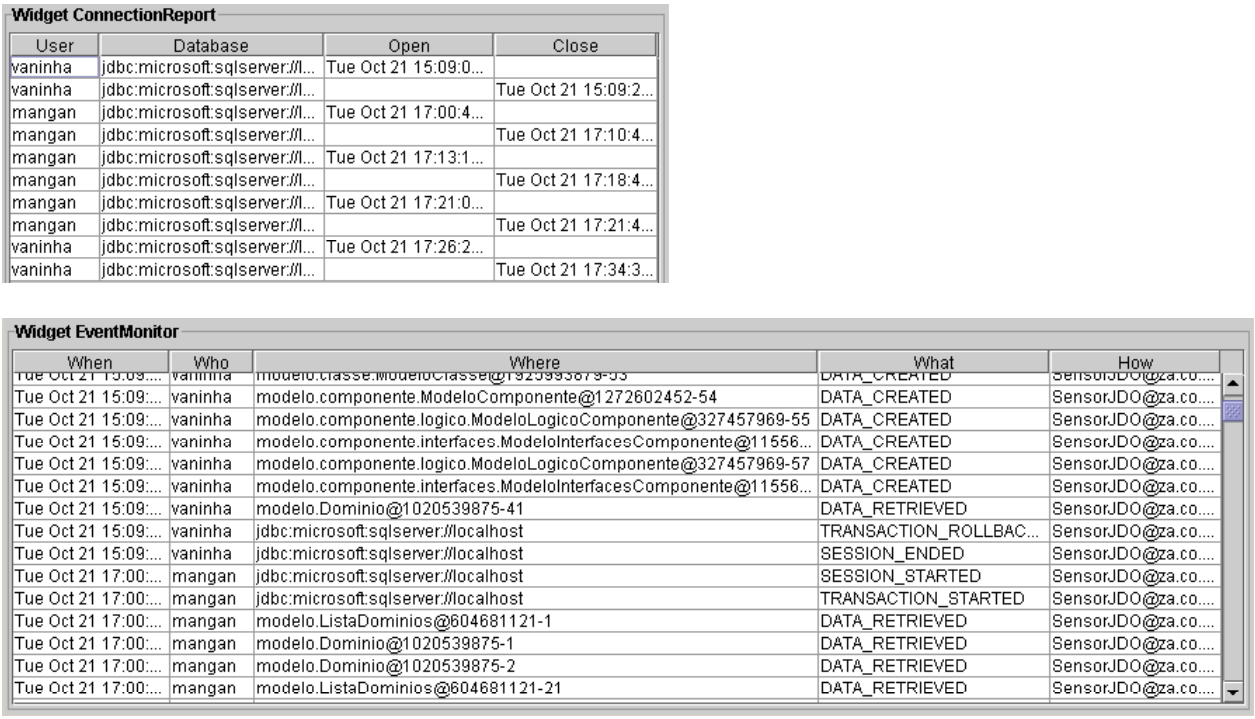
```
-- modelo.ModeloAbstrato
create table modelo_abstrato (
    modelo_abstrato_id INTEGER not null,    -- <pk>
    jdo_class INTEGER not null,             -- <class-id>
    alocado SMALLINT,                       -- alocado
    criado SMALLINT,                        -- criado
    descricao VARCHAR(255),                 -- descricao
    dono VARCHAR(255),                      -- dono
    esteriotipo VARCHAR(255),               -- esteriotipo
    identificador INTEGER,                  -- identificador
    modificado SMALLINT,                    -- modificado
    nome VARCHAR(255),                      -- nome
    novo SMALLINT,                          -- novo
    pai_modelo_abstrato_id INTEGER,         -- pai
    saved SMALLINT,                         -- saved
    jdo_version SMALLINT not null,          -- <opt-lock>
    serializable_list_model_id INTEGER,     -- Aplicacao.contextos
    serializable_list_model_id2 INTEGER,    -- Dominio.aplicacoes
    carregado SMALLINT,                     -- Dominio.carregado
    nome_base VARCHAR(255),                 -- Dominio.nomeBase
    modelo_processo INTEGER,                -- InfraEstrutura.modeloProcesso
    diagrama_padrao_id INTEGER,             -- Diagrama.diagrama
    classificadora SMALLINT,                -- LigAssociacaoTipoNegocio.classificadora
    derivada SMALLINT,                      -- LigAssociacaoTipoNegocio.derivada
    multiplicidade_destino VARCHAR(255),    -- LigAssociacaoTipoNegocio.multiplicidadeDestino
    multiplicidade_origem VARCHAR(255),     -- LigAssociacaoTipoNegocio.multiplicidadeOrigem
    navegavel_destino SMALLINT,             -- LigAssociacaoTipoNegocio.navegavelDestino
```

**Figura 28 – *Script* de criação de esquema gerado pelo JDO *Genie* para o *OdysseyShare***

O passo seguinte foi acoplar o Sensor, implementado como aspecto, à implementação JDO Genie. Para isso, utilizou-se a ferramenta AspectJ sobre os aspectos *AbstractSensor* e *SensorJDO* contra o arquivo *jdogenie.jar* (que contém todas as classes da implementação JDO Genie), gerando um novo arquivo que denominamos *arngenie.jar*. O processo de pós-compilação do AspectJ copia apenas os arquivos *.class* de *jdogenie.jar* para *arngenie.jar*, tendo sido necessário copiar, manualmente, os arquivos não .class para que a JDO Genie continuasse funcionando normalmente.

Após a geração da versão monitorada do JDO Genie, a única modificação efetuada no *OdysseyShare*, para ativar o monitoramento dos eventos, foi alterar seu *script* de execução para que o *classpath* apontasse para o arquivo *arngenie.jar* ao invés do anterior *jdogenie.jar*.

Nesse ponto, o *OdysseyShare* já pode ser executado com o serviço de percepção ativado. Para isso, basta instanciar a aplicação *RMIRegistry* e o servidor de percepção de Ariane em uma máquina virtual Java (JVM) e a aplicação *OdysseyShare* em uma outra JVM (fisicamente, no entanto, podem ser executados em uma mesma máquina).



**Widget ConnectionReport**

| User | Database | Open | Close |
|---|---|---|---|
| vaninha | jdbc:microsoft:sqlserver://l... | Tue Oct 21 15:09:0... | |
| vaninha | jdbc:microsoft:sqlserver://l... | | Tue Oct 21 15:09:2... |
| mangan | jdbc:microsoft:sqlserver://l... | Tue Oct 21 17:00:4... | |
| mangan | jdbc:microsoft:sqlserver://l... | | Tue Oct 21 17:10:4... |
| mangan | jdbc:microsoft:sqlserver://l... | Tue Oct 21 17:13:1... | |
| mangan | jdbc:microsoft:sqlserver://l... | | Tue Oct 21 17:18:4... |
| mangan | jdbc:microsoft:sqlserver://l... | Tue Oct 21 17:21:0... | |
| mangan | jdbc:microsoft:sqlserver://l... | | Tue Oct 21 17:21:4... |
| vaninha | jdbc:microsoft:sqlserver://l... | Tue Oct 21 17:26:2... | |
| vaninha | jdbc:microsoft:sqlserver://l... | | Tue Oct 21 17:34:3... |

**Widget EventMonitor**

| When | Who | Where | What | How |
|---|---|---|---|---|
| Tue Oct 21 15:09:... | vaninha | modelo.componente.ModeloComponente@1272602452-54 | DATA_CREATED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | modelo.componente.logico.ModeloLogicoComponente@327457969-55 | DATA_CREATED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | modelo.componente.interfaces.ModeloInterfacesComponente@11556... | DATA_CREATED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | modelo.componente.logico.ModeloLogicoComponente@327457969-57 | DATA_CREATED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | modelo.componente.interfaces.ModeloInterfacesComponente@11556... | DATA_CREATED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | modelo.Dominio@1020539875-41 | DATA_RETRIEVED | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | jdbc:microsoft:sqlserver://localhost | TRANSACTION_ROLLBAC... | SensorJDO@za.co.... |
| Tue Oct 21 15:09:... | vaninha | jdbc:microsoft:sqlserver://localhost | SESSION_ENDED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | jdbc:microsoft:sqlserver://localhost | SESSION_STARTED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | jdbc:microsoft:sqlserver://localhost | TRANSACTION_STARTED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | modelo.ListaDominios@604681121-1 | DATA_RETRIEVED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | modelo.Dominio@1020539875-1 | DATA_RETRIEVED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | modelo.Dominio@1020539875-2 | DATA_RETRIEVED | SensorJDO@za.co.... |
| Tue Oct 21 17:00:... | mangan | modelo.ListaDominios@604681121-21 | DATA_RETRIEVED | SensorJDO@za.co.... |

**Figura 29 – Visualização de eventos produzidos pelo monitoramento do *OdysseyShare***

Para exibição dos eventos de percepção gerados por Ariane, foram criados dois componentes visuais: o Histórico de Conexão (que exibe apenas eventos de sessão) e o Monitor de Eventos (que exibe todos os eventos). Esses componentes apresentam as informações de percepção em uma tabela, que é alimentada, sincronamente, sempre que um novo evento é produzido no servidor de percepção. Além disso, ao serem iniciados, esses componentes fazem consulta ao servidor de percepção para recuperar os eventos passados, armazenados no BD de Eventos. A Figura 29 mostra esses componentes, alimentados com eventos gerados pelo monitoramento de ações sobre o *OdysseyShare*. Percebe-se que essa forma de visualização dos eventos, para o usuário final, causa

sobrecarga de informações, e acaba representando o mesmo que nenhuma informação. Assim, diferentes formas de visualização dos dados faz-se necessária.

Porém, os eventos produzidos por Ariane servem como fonte de dados possibilitando a criação de diferentes componentes visuais de percepção. Os mesmos eventos, contendo a mesma informação, podem ser visualizados de diferentes maneiras, visando facilitar o consumo e compreensão das ações ocorridas, considerando-se as necessidades dos vários usuários do grupo. O trabalho dos desenvolvedores desses componentes diminui bastante, uma vez que eles precisam se concentrar, apenas, nas questões relacionadas à apresentação das informações .

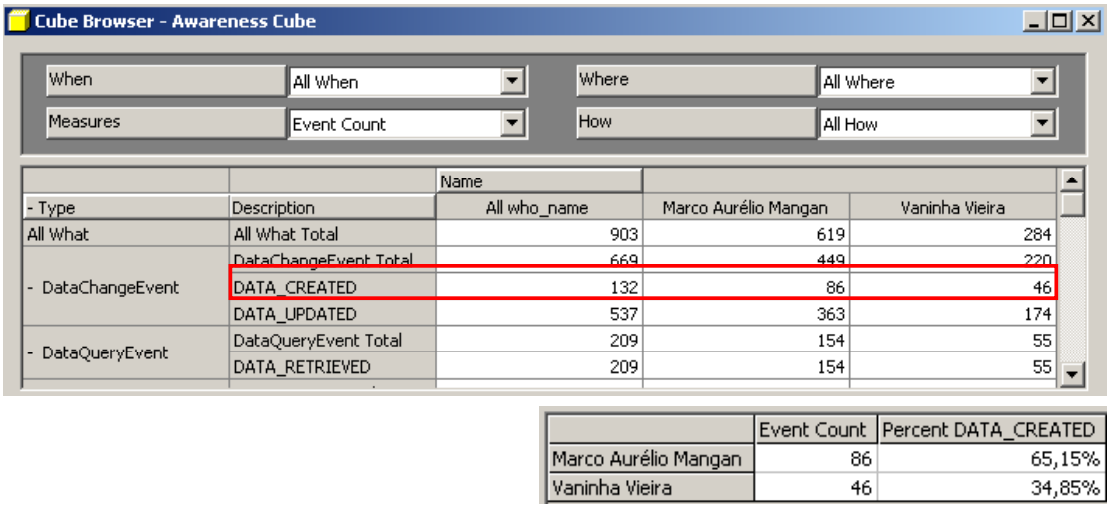



**Figura 30 – Visualização dos Dados do Cubo de Percepção**

Uma outra forma de visualizar e analisar esses eventos, disponibilizada por Ariane, é através do cubo de percepção. A Figura 30 apresenta um exemplo de consulta aos dados do cubo de percepção. Na parte superior da tela aparecem as dimensões definidas. Pode-se utilizar filtros sobre qualquer das dimensões e cruzar os dados de qualquer das dimensões. No exemplo, a visualização dos dados confronta as dimensões *What*, que indica as ações efetuadas, com a dimensão *Who*, que indica quem executou cada tipo de ação. Por essa visualização é possível verificar quais os usuários mais participativos para cada um dos tipos de ações considerados. Diferentes visualizações das consultas podem ser feitas, como no exemplo, uma amostra percentual das contribuições dos usuários para a criação de artefatos (*DATA_CREATED*). Mudanças na definição das dimensões, adicionando e removendo hierarquias, podem ser feitas através da própria ferramenta OLAP.

Para geração do cubo, foi executado o pacote DTS, definido em Ariane. Para analisar os dados gerados no cubo de percepção foi utilizada a ferramenta *Microsoft Analysis Services* (MICROSOFT, 2003a). Existem diversas outras ferramentas OLAP disponíveis. Os principais SGBDs já vêm disponibilizando suas próprias ferramentas OLAP, como o MetaCube (Informix) ou o Discoverer (Oracle) (CAMPOS, 2000). Uma ferramenta brasileira bastante utilizada é a Maestro (CAMPOS, 2000). Entre as de código aberto, pode-se citar o JPivot que atua junto ao servidor OLAP Mondrian (MONDRIAN TEAM, 2003).

## IV.5. DISCUSSÃO SOBRE POTENCIAIS UTILIZAÇÕES DE ARIANE

O mecanismo de percepção Ariane funciona como uma infra-estrutura de percepção, que visa ampliar a oferta de informações de percepção de aplicações cooperativas a partir do monitoramento do processo de persistência dessas aplicações. Os tipos de eventos gerados por Ariane possibilitam que os mesmos sejam utilizados de diversas maneiras para promover a percepção, como sumarizado na Tabela 2.

**Tabela 2 – Utilização dos Eventos Gerados por Ariane**

| CATEGORIA | POTENCIAIS UTILIZAÇÕES |
|---|---|
| Sessão | Monitorar tempo de conexão dos usuários; agrupar seqüências de operações realizadas por um usuário em uma sessão de trabalho. |
| Transação | Monitorar operações finalizadas com sucesso ou não concluídas, devido a falhas; agregar bloco de operações por transação. |
| Mudança (Esquema) | Monitorar mudanças realizadas sobre o esquema da base de dados (interessa, particularmente, aos desenvolvedores das aplicações cooperativas). |
| Mudança (Dados) | Monitorar mudanças sobre os dados armazenados na base de dados (permite identificar, por exemplo, os usuários mais participativos dentro do grupo). |
| Consulta | Monitorar consultas executadas sobre o esquema ou os dados armazenados na base de dados (pode indicar, por exemplo, se todos os usuários já leram uma informação, ou pode indicar um conjunto de artefatos que interessam aos mesmos usuários). |
| Exceção | Monitorar ocorrência de erros quando da execução de ações sobre a base de dados (pode indicar conflitos que devam ser resolvidos pelo coordenador do grupo). |

Por exemplo, através do monitoramento das operações de iniciar e finalizar sessão ao SGBD, é possível conhecer o período de atividade dos usuários. Com essa informação, analistas ou coordenadores do grupo podem perceber que determinados indivíduos costumam estar conectados em intervalos de tempo comuns, indicando que essas pessoas são candidatas a colaborar na execução de alguma tarefa.

Em termos gerais, o monitoramento e manutenção das ações executadas sobre os artefatos persistentes da aplicação apresentam diversas vantagens, permitindo que usuários:

- Localizem onde determinadas atividades ocorreram;
- Reconstruam atividades passadas, próprias ou de outros usuários, auxiliando o entendimento do estado atual de um artefato;
- Identifiquem a seqüência de passos executada para que um artefato chegue ao seu estado atual. O SGBD armazena apenas o estado atual de um artefato. Sem o monitoramento os estados intermediários são perdidos;
- Resolvam conflitos. O monitoramento permite identificar ações realizadas por um usuário que não puderam ser efetivadas devido a conflitos ou bloqueios. O monitoramento permite identificar os usuários que estavam utilizando o mesmo artefato, permitindo que eles entrem em contato para resolver conflitos.

Existem aplicações onde o monitoramento, associado ao uso de métricas e de ferramentas de análise, pode ser bastante útil. Em aplicações que apóiem o desenvolvimento colaborativo de software, pode auxiliar a conhecer a maneira como as pessoas trabalham, permitindo identificar o ritmo de trabalho dos usuários e, baseado no aprendizado sobre as práticas atuais do grupo, especialistas podem definir formas e métodos de trabalho que disciplinem e uniformizem a maneira como as pessoas devem trabalhar. A idéia de utilizar práticas estabelecidas para disciplinar o processo de desenvolvimento de *software* faz parte do chamado **Processo de Software Personalizado** (PSP – Personal Software Process) (HUMPHREY, 2000).

Em **aplicações cientes de contexto** (*context-aware*), o monitoramento auxilia a identificação de contextos. Contextos são informações utilizadas para caracterizar a situação de uma entidade (pessoa, local, artefato) relevantes para a interação de um usuário e uma aplicação (DEY, 2001). Aplicações baseadas em contexto podem utilizar esses contextos para prover informações e/ou serviços relevantes para seus usuários, onde relevância depende da tarefa do usuário (DEY, 2001);

Em **aplicações de gerência de configuração** que desejam, não apenas, controlar o estado final de um artefato (após o usuário dar o *check in* e atualizar o artefato no repositório), mas acompanhar as alterações que estão sendo realizadas sobre os artefatos enquanto o usuário detém o controle e está trabalhando sobre ele.

# *Capítulo V*

# *Conclusão*

## V.1. CONSIDERAÇÕES FINAIS

Nesta dissertação, foi destacada a importância da percepção para apoiar a atividade colaborativa. Sem o conhecimento do que ocorre no contexto do grupo e do que os demais colaboradores estão fazendo, o trabalho individual fica bastante prejudicado, gerando uma sensação de solidão e isolamento que vai de encontro aos princípios do trabalho cooperativo. SGBDs são comumente utilizados para persistir as informações produzidas pelo grupo em aplicações cooperativas, porém SGBDs são sistemas de propósito geral, não tendo sido desenvolvidos pensando especificamente nos requisitos das aplicações cooperativas. Assim, os SGBDs atuais não oferecem suporte à percepção. Acredita-se que o monitoramento das ações produzidas no processo normal de persistência das aplicações cooperativas pode contribuir para ampliar a oferta de informações de percepção dessas aplicações.

O apoio à percepção em aplicações cooperativas ainda é um tópico em aberto. Essas aplicações, em geral, oferecem pouco ou nenhum serviço que apóie a percepção do que mudou nos artefatos compartilhados para apoiar, especialmente, as interações assíncronas. A maioria dos mecanismos de percepção existentes está fortemente acoplada a uma aplicação cooperativa particular. Alguns trabalhos, na literatura também apostam nas vantagens em oferecer informações de percepção sobre aplicações cooperativas, de forma genérica, atendendo a um grande número de aplicações. Em especial, destacam-se os sistemas SISCO (MARIANI, 1997), Nessie (PRINZ, 1999) e BW (PINHEIRO, 2001).

Entretanto, o objetivo desta dissertação consistiu em oferecer um mecanismo de percepção que seja independente, ao mesmo tempo, da aplicação cooperativa e do SGBD utilizado por essa aplicação. Além disso, objetivava-se facilitar o uso das informações de percepção coletadas. Nesse sentido, os trabalhos da literatura, ou são

muito dependentes e acoplados à aplicação, ou são acoplados ao SGBD. Além disso, eles não se preocupam com o valor agregado das informações de percepção, que pode ser obtido através de ferramentas de processamento analítico.

Para atender a esses objetivos, alguns desafios tecnológicos tiveram que ser transpostos. De início, a flexibilidade de um SGBD está limitada pelo fato do SGBD não reconhecer o uso colaborativo. Do outro lado, buscou-se um desenvolvimento o mais independente possível da plataforma na medida em que utilizou tecnologias abertas e padronizadas como base.

Assim, foi projetado um mecanismo de percepção, denominado Ariane, onde se destaca o uso das tecnologias de Programação Orientada a Aspectos (AOP – *Aspect Oriented Programming*), *Java Data Objects* (JDO) e ferramentas de processamento analítico (OLAP - *Online Analytical Processing*).

Dentre os requisitos estabelecidos para Ariane o principal deles é que a solução não fosse limitada a uma aplicação ou banco de dados específico e que, dessa maneira, não fosse necessário modificar a estrutura interna de um SGBD ou de uma aplicação para que o mecanismo pudesse ser utilizado. Além disso, estabeleceu-se que todas as interações realizadas sobre a base de dados fossem persistidas, produzindo uma memória do grupo, utilizando como contexto o modelo de representação da aplicação cooperativa e não do modelo de persistência do SGBD. Um outro requisito é permitir a análise efetiva das informações de percepção.

Ariane utiliza a abordagem de percepção baseada em notificação de eventos e seu processo de percepção consiste de quatro etapas principais: produção, distribuição, consumo e análise de eventos, onde cada tipo de evento corresponde a um tipo de ação que pode ser executada por uma aplicação cooperativa sobre um SGBD, na persistência dos seus artefatos. Essa divisão também foi utilizada para montar a arquitetura do mecanismo. Foi projetado e implementado um protótipo de Ariane, de modo a analisar a viabilidade de implementação da proposta.

A produção de eventos ocorre através de sensores, acoplados ao mecanismo de persistência utilizado pela aplicação cooperativa. Para tornar Ariane independente de aplicação e SGBD, optou-se pelo monitoramento do mecanismo de persistência JDO, que é uma proposta de padrão para persistência transparente em aplicações Java. A implementação dos sensores no protótipo de Ariane foi realizada através do novo

paradigma de desenvolvimento de aplicações, a programação orientada a aspectos, que permite encapsular requisitos transversais de uma aplicação (aqueles que aparecem repetidos em diferentes pontos da aplicação), através de uma nova unidade de programação: os aspectos. Dessa maneira, os sensores foram implementados como aspectos e acoplados a implementações de mecanismos de persistência que utilizam a especificação padronizada JDO.

As informações coletadas pelos sensores são enviadas a um servidor de percepção, gerando eventos que são distribuídos a componentes de apresentação da percepção, registrados. Os eventos são representados de forma estruturada, o que facilita sua utilização e interpretação por diferentes componentes de apresentação. A memória das interações do grupo é mantida através do armazenamento histórico dos eventos. Foi realizada uma modelagem multidimensional sobre os eventos armazenados, através de uma estrutura denominada cubo de percepção, possibilitando que as informações de percepção produzidas sejam analisadas, através de ferramentas OLAP, apoiando usuários do grupo, com papel de analista, a compreender as interações dos participantes e os auxiliando em processos de tomada de decisão.

Uma análise de uso desse protótipo foi feita junto ao ambiente *OdysseyShare*. Para isso, foi incluído o JDO como mecanismo de persistência nesse ambiente (ampliando as suas opções de persistência, que sempre foi uma preocupação). Nenhuma implementação específica para acoplamento do mecanismo Ariane ao *OdysseyShare* foi necessária, para que o monitoramento das ações sobre essa aplicação fosse realizada. Foram implementados alguns componentes de percepção para avaliar o modelo de notificação de eventos. Um conjunto de ações foi monitorada para geração do cubo de percepção, sobre o qual foi utilizada uma ferramenta OLAP, para verificação da viabilidade de realizar consultas analíticas sobre o modelo multidimensional proposto.

O protótipo de Ariane foi desenvolvido pensando em aplicações escritas na linguagem Java que utilizem o JDO como mecanismo de persistência transparente. O uso de Ariane visa apoiar o trabalho de desenvolvedores de aplicações cooperativas e de componentes visuais de percepção. Com Ariane, o esforço dos desenvolvedores para monitorar uma outra aplicação Java que utilize JDO é zero, nenhuma programação adicional precisa ser feita. Da mesma maneira, se a aplicação desejar trocar de SGBD, o monitoramento também não é afetado. Caso o desenvolvedor deseje construir um novo componente visual de percepção, ele necessita apenas se conectar ao servidor de

percepção de Ariane e implementar o tratamento visual dos eventos. Se o desenvolvedor deseja utilizar Ariane para monitorar uma outra aplicação que não utilize o JDO como mecanismo de persistência, ele terá que construir um novo sensor, que herde de *AbstractSensor* e indique os pontos de acesso da nova aplicação, e colete as informações 5W+1H (ou um conjunto delas). Supondo que o desenvolvedor deseje modificar a granularidade dos eventos, passando a considerar como eventos de percepção outros tipos que não os definidos no projeto de Ariane, o que deverá ser feito é a extensão do modelo de eventos de Ariane e a criação de um novo sensor, que monitore e colete esses eventos. A AOP facilita sobremaneira a definição e implementação desses novos sensores.

## V.2. CONTRIBUIÇÕES

Dentre as contribuições desta dissertação, podemos citar:

(i) A proposta de uma abordagem de percepção baseada no monitoramento de bases de dados compartilhadas, aproveitando o processo usual de persistência das aplicações cooperativas;

(ii) A modelagem dos eventos, segundo a representação 5W+1H;

(iii) A extensão ao processo de percepção proposto em (SOHLENKAMP, 1998), incluindo a etapa de análise das informações de percepção produzidas. Essa extensão possibilitou que ferramentas de análise fossem utilizadas sobre o histórico de informações de percepção do grupo para aquisição de valor agregado, aumentando o conhecimento e compreensão sobre as interações do grupo (escondidas em um grande volume de informações), e auxiliando usuários do grupo com papel de analista em atividades de tomada de decisão;

(iv) A modelagem multidimensional para os eventos produzidos e a geração de um cubo de percepção alimentado com os eventos históricos produzidos. A geração desse cubo possibilitou que o registro histórico dos eventos produzidos fosse utilizado para análises em ferramentas OLAP;

(v) projeto e implementação de Ariane, um mecanismo de percepção flexível e modular, baseado em tecnologias padronizadas e abertas (JDO, AspectJ, plataforma Java), que pode ser integrado a qualquer aplicação que utilize o JDO

como mecanismo de persistência, e que possui serviços de extensão que permite que qualquer tipo de ação executada por qualquer aplicação, seja monitorado, de forma simplificada e não intrusiva (sem interferência direta na aplicação);

(vi) A utilização do AspectJ para monitorar implementações JDO e o mapeamento das ações do JDO que afetam o estado interno de um banco de dados. Essas escolhas tecnológicas possibilitaram que o mecanismo de percepção fosse efetivamente independente de aplicação ou SGBD específicos, podendo ser utilizado por um grande número de aplicações (desenvolvidas na plataforma Java), acessando um grande número de SGBDs (as implementações JDO existentes compreendem os principais SGBDs existentes, tanto comerciais quanto de código aberto). O uso do AspectJ, trouxe como vantagens, também, a independência de uma implementação JDO específica (atuando no âmbito da especificação JDO);

(vii) A utilização do protótipo de Ariane junto ao ambiente OdysseyShare, ampliando as ofertas de percepção assíncrona desse ambiente. O acoplamento de Ariane ao OdysseyShare foi feito de forma totalmente transparente para a aplicação.

## V.3. LIMITAÇÕES E TRABALHOS FUTUROS

Esta dissertação apresenta, ainda, algumas limitações, descritas a seguir, bem como alguns trabalhos que podem ser realizados futuramente, dando continuidade à pesquisa ora realizada:

(i) A questão "por que" (*why*), que indica os motivos por trás da realização de uma ação, não foi tratada nesta dissertação, devido às dificuldades apresentadas. Entretanto, acredita-se que a resposta a essa pergunta é uma das mais importantes, para entender e aceitar as ações de um usuário e para resolver conflitos. Poderiam ser utilizadas anotações sobre os eventos ocorridos, ou investigadas técnicas de inteligência artificial que permitissem inferir os motivos a partir de seqüências de ações. Acredita-se que, a partir de informações sobre o processo de trabalho dos usuários, seja possível inferir informações sobre essa questão;

(ii) Também a questão "como"(*how*) não foi resolvida satisfatoriamente. Imagina-se que o como indique o processo de realização da ação. No entanto, a maneira como Ariane monitora os eventos não permite que esse tipo de informação seja obtida.

Deve-se estudar formas de como identificar essa questão. Imagina-se, também, que a partir do processo de trabalho dos usuários, essa informação possa ser obtida;

(iii) Questões específicas de apresentação e filtragem das informações foi deixada a cargo dos componentes visuais de percepção. No entanto, acreditamos que novos filtros poderiam ser embutidos em Ariane, para tornar a coleta e distribuição dos eventos mais seletiva. Poderia-se utilizar, por exemplo, filtros com aprendizado, que modificassem sua seletividade a partir das consultas mais realizadas;

(iv) Técnicas de mineração de dados poderiam ser utilizadas sobre o histórico de eventos para descoberta de conhecimento e padrões escondidos;

(v) O protótipo de Ariane foi avaliado, apenas, junto no ambiente *OdysseyShare*. Deseja-se criar outros componentes visuais de percepção, para validar os eventos coletados, e testar o uso desses componentes em outros ambientes;

(vi) O monitoramento dos eventos em Ariane, atualmente, não leva em consideração o contexto onde as ações ocorreram. Assim, informações do tipo, qual a aplicação utilizada pelo usuário, ou em que tela o usuário estava ou qual tarefa estava sendo desenvolvida, por exemplo, são desvinculadas dos eventos. Imagina-se que tratar o contexto das ações pode ser bastante importante para o entendimento das mesmas.

# REFERÊNCIAS BIBLIOGRÁFICAS


APACHE, 2003, "Proposal for a OJB based JDO(R) implementation". In: http://db.apache.org/ojb/jdo-proposal.html, Acessado em 10/2003.

ARAÚJO, R. M., 2000, *Ampliando a Cultura de Processos de Software - Um Enfoque Baseado em Groupware e Workflow*, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro.

ASPECTJ TEAM, 2003, "AspectJ Project Home Page". In: http://www.aspectj.org, Acessado em 10/2003.

ASSIS, R. L., 2000, *Facilitando a percepção em ambiente virtuais de aprendizado através da tecnologia groupware*, M.Sc., Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil.

BORGES, M. R. S., PINO, J. A., 1999, "Awareness Mechanisms for Coordination in Asynchronous CSCW". In: *9th Workshop on Information Techonologies and Systems*, pp. 69-74, Charlotte, North Carolina.

BORGES, M. R. S., PINO, J. A., 2000, "Requirements for Shared Memory in CSCW Applications". In: *Proceedings of the 10th Workshop on Information Techonologies and Systems*, pp. 211-216, Australia.

BORGES, M. R. S., PINO, J. A., SALGADO, A. C., 2000, "Requirements for Shared Memory in CSCW Applications". In: http://equipe.nce.ufrj.br/mborges/publicacoes/DBSCWCOOPIS.doc, Acessado em 10/2003.

BÜRGER, M., 1999, *Support of awareness for group work with shared workspaces*, M.Sc., UTZ Verlag, München.

CADIZ, J. J., FUSSEL, S. R., KRAUT, F., et al., 1998, "The Awareness Monitor: A Coordination Tool for Asynchronous, Distributed Work Teams". In: http://research.microsoft.com/~jjcadiz/, Acessado em 10/2003.

CAMPOS, M. L. M., 2000, "Data Warehouse". In: *Anais XV SBBD / XIV SBES - Mini-Cursos / Tutoriais*, pp. 66-120, João Pessoa - PB - Brasil.

CERI, S., COCHRANE, R. J., WIDOM, J., 2000, "Practical Applications of Triggers and Constraints: Successes and Lingering Issues". In: *Proc. of the 26th VLDB.*, pp. 254-262, Cairo, Egypt.

CHAVEZ, C. F. G., LUCENA, C. J. P., 2001, "Design-level Support for Aspect-oriented Software Development". In: *Proc. of the Workshop on Advanced Separation of Concerns in Object-oriented Systems (ASoC) at OOPSLA'2001*, Tampa Bay, Florida, USA.

CVS, 2003, "Concurrent Version System Home Page". In: http://www.cvshome.org, Acessado em 10/2003.

DATAWARE, 2002, "Grupo DatAware - HomePage". In: http://genesis.nce.ufrj.br/dataware, Acessado em 10/2003.

DAVID, J. M. N., BORGES, M. R. S., 2001, "Selectivity of Awareness Componentes in Asynchronous CSCW Environments". In: *Proc. of 7th International Workshop on Groupware (CRIWG'01)*, pp. 115-124, Darmstadt, Germany.

DEY, A. K., 2001, "Understanding and Using Context". In: http://citeseer.nj.nec.com/dey01understanding.html, Acessado em 10/2003.

DIAS, M. S., 1998, *COPSE: Um ambiente de suporte ao projeto cooperativo de software*, M.Sc., COPPE/UFRJ, Rio de Janeiro, Brasil.

DOURISH, P., 1997, "Extending awareness beyond synchronous collaboration". In: *Proc. Workshop on Awareness in Collaboration Systems (CHI'97)*, Atlanta, USA, In: http://www.best.com/~jpd/chi97-awareness.html, Acessado em 10/2003.

DOURISH, P., BELLOTTI, V., 1992, "Awareness and Coordination in Shared Workspaces". In: *Proc. ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pp. 107-114, Toronto, Ontario.

ELLIS, L., GIBBS, S. J., REIN, G. L., 1991, "Groupware: Some Issues and Experiences", *Communications of the ACM*, v. 34, n. 1, pp. 38-58.

FUSSEL, S. R., PANKOKE-BABATZ, U., PRINZ, W., 1995, "Supporting cooperative awareness with local event mechanisms: The GroupDesk system". In: *Proceedings of ECSCW'95*, pp. 247-262, Stockholm, Sweden.

GARCIA, A., SILVA, V., CHAVEZ, C., et al., 2002, "Engineering Multi-Agent Systems with Aspects and Patterns", *Journal of the Brazilian Computer Society*, v. 8, n. 1, pp. 57-72.

GELLERSEN, H. W., BEIGL, M., 1999, "Ambient Telepresence: Colleague Awareness in Smart Environments". In: *Proc. Managing Interactions in Smart Environments (MANSE'99)*, pp. 80-88, Ireland, Springer-Verlag London.

GEROSA, M. A., FUKS, H., LUCENA, C. J. P., 2001, "Elementos de percepção como forma de facilitar a colaboração em cursos via Internet". In: *Anais XII Simpósio Brasileiro de Informática na Educação - SBIE*, pp. 194-202, Vitória - ES - Brasil.

GMD, 2002, "BSCW". In: http://bscw.gmd.de, Acessado em 03/2002.

GREIF, I., SARIN, S., 1987, "Data Sharing in Group Work", *ACM Transactions on Office Information Systems*, v. 2, n. 5, pp. 187-211.

GUTWIN, C., 1997, *Workspace Awareness in Real-Time Distributed Groupware*, Ph.D., University of Calgary, In: http://www.cs.usask.ca/faculty/gutwin/publications/, Acessado em 10/2003.

GUTWIN, C., GREENBERG, S., 1996, "Workspace Awareness for Groupware". In: *Proc. Conference on Human Factors in Computing Systems*, pp. 208-209, Vancouver.

GUTWIN, C., GREENBERG, S., 2002, "A Descriptive Framework of Workspace Awareness for Real-Time Groupware". In: *Computer Supported Cooperative Work*, v. 11(3-4), pp. 411-446, Special Issue on Awareness in CSCW, Kluwer Academic Press.

HEMISPHERE, 2003, "JDO Genie". In: http://www.hemtech.co.za/jdo/index.html, Acessado em 10/2003.

HO, W. M., PENNANEAC'H, F., JEZEQUEL, J. M., et al., 2000, "Aspect-Oriented Design with the UML". In: *Proc. Multi-Dimensional Separation of Concerns Workshop (ICSE'00)*, Limerick, Ireland.

HUMPHREY, W. S., 2000, "The Personal Software Process (PSP)". In: http://www.sei.cmu.edu/publications/documents/00.reports/00tr022.html, Acessado em 10/2003.

ICQ, 2003, "ICQ". In: http://www.icq.com, Acessado em 10/2003.

INMON, W. H., 1997, *Como Construir o Data Warehouse*. Tradução da 2ª ed. ed., Rio de Janeiro - RJ, Editora Campus.

JDO CENTRAL, 2003, "Developers Community for Java Data Objects". In: http://www.jdocentral.com, Acessado em 10/3 A.D.

JDO COMMUNITY, 2003, "Grupo de Discussão sobre JDO". In: http://groups.yahoo.com/group/JavaDataObjects/, Acessado em 10/2003.

KICZALES, G., HILSDALE, E., HUGUNIN, J., et al. , 2001, "An Overview of AspectJ", *Lecture Notes in Computer Science*, v. 2072, pp. 327-355.

KICZALES, G., LAMPING, J., MENHDHEKAR, A., et al., 1997, "Aspect Oriented Programming". In: *Proc. European Conference on Object-Oriented Programming (ECOOP'97)*, v. 1241 of LNCS, pp. 220-242, Springer-Verlag.

KIMBALL, R., MERZ, R., 2000, *The Data WebHouse Toolkit*, New York, USA, John Wiley & Sons, Inc.

KREIJNS, K., KIRSCHNER, P. A., 2001, "The Social Affordances of Computer-Supported Collaborative Learning Environments". In: *Proc. 31th ASEE/IEEE Frontiers in Education Conference*, pp. 12-17, Reno, NV.

LIBELIS, 2003, "LiDO JDO". In: http://www.libelis.com, Acessado em 06/2003.

LIECHTI, O., 2000, "Awareness and the WWW: an Overview". In: *Proc. Workshop on 'Awareness and the WWW' (CSCW '00)*, v. 21 (3), pp. 3-12, ACM Press, New York, USA.

LIU, L., PU, C., BARGA, R., et al., 1996, "Differential Evaluation of Continual Queries". In: *Proc. 16th International Conference on Distributed Computing Systems*, pp. 458-465, Hong Kong.

MANGAN, M. A. S., ARAÚJO, R. M., KALINOWSKI, M., et al., 2002, "Towards the Evaluation of Awareness Information Support Applied to Peer Reviews of Software Engineering Diagrams". In: *Proc. 7th International Conference on CSCW in Design (CSCWD'02)*, pp. 49-54, Rio de Janeiro, Brasil.

MARIANI, J. A., 1997, "SISCO: Providing a Cooperation Filter for a Shared Information Space". In: *Proceedings of the International ACMSIGGROUP Conference on Supporting Group Work: The Integration Challenge*, pp. 376-384, Phoenix, Arizona, USA, New York: ACM Press.

MARIANI, J. A., RODDEN, T., 1991, "The impact of CSCW on database technology". In: *Proceedings of the IFIP International Workshop on CSCW*, pp. 146-161, Berlin, Germany.

MATTOSO, M. L. Q., CAVALCANTI, M., SOUZA, R., et al., 2002, "Gerência de Documentos XML no GOA". In: *SBES, Ferramentas*, pp. 402-407, Gramado, Brasil.

MCCAFFREY, L., 1998, *Representing Change in Persistent Groupware Environments*. Grouplab report. Department of Computer Science, University of Calgary, Alberta, Canada, January.

MICROSOFT, 2003a, "Microsoft Analysis Services 2000 - Documentação". In: http://www.microsoft.com/sql/olap, Acessado em 10/2003a.

MICROSOFT, 2003b, "Microsoft NetMeeting", Acessado em 07/2003b.

MOLLI, P., SKAF-MOLLI, H., OSTER, G., et al., 2002, "SAMS: Synchronous, Asynchronous, Multi-Synchronous Environments". In: *Proc. 7th International Conference on CSCW in Design (CSCWD'02)*, pp. 80-84, Rio de Janeiro, Brasil.

MONDRIAN TEAM, 2003, "Mondrian". In: http://mondrian.sourceforge.net, Acessado em 10/2003.

OBJECTDB, 2003, "ObjectDB for Java/JDO". In: http://www.newfreeware.com/development/431/, Acessado em 10/2003.

ÖZSU, M. T., VALDURIEZ, P., 1999, *Principles of Distributed Database Systems*. 2nd ed., New Jersey, USA, Prentice-Hall.

P6SPY TEAM, 2003, "P6Spy Project Home Page". In: http://sourceforge.net/projects/p6spy/, Acessado em 10/2003.

PARSOWITH, S., FITZPATRICK, G., KAPLAN, S., et al., 2003, "TickerTape: notification and communication in a single line". In: http://www.dstc.edu.au/Research/Projects/EWP/Papers/apchi98.html, Acessado em 10/2003.

PEREIRA, S. L., 2002, "AspectJ - Slides (Seminário)". In: www.ime.usp.br/~kon/MAC5715/aulas/slides/AspectJ.pdf , Acessado em 10/2003.

PETERSON, T., 2000, *Microsoft SQL Server 2000 Data Transformation Services (DTS)*. 1st ed., SAMS.

PINHEIRO, M. K., 2001, *Mecanismo de Suporte à Percepção em Ambientes Cooperativos*, Dissertação de M.Sc., PPGC/UFRGS, Porto Alegre, RS, Brasil.

PINHEIRO, M. K., LIMA, J. V., BORGES, M. R. S., 2002, "A Framework for Awareness Support in Groupware Systems". In: *Proc. 7th International Conference on CSCW in Design (CSCWD'02)*, pp. 13-18, Rio de Janeiro, Brasil.

PRAKASH, A., SHIM, H. S., LEE, J. H., 1999, "Data Management Issues and Tradeoffs in CSCW Systems", *IEEE Transactions on Knowledge and Data Engineering*, v. 11, n. 1, pp. 213-227.

PREGUIÇA, N., MARTING, J. L., DOMINGOS, H., et al., 2000, "Data Management Support for Asynchronous Groupware". In: *Proceedings of the 2000 ACM Conference on Computer-Supported Cooperative Work*, pp. 68-78, Philadelphia, PA, USA.

PRESSMAN, R. S., 1999, "Project Management Concepts", *Software Engineering - A Practitioner's Approach*, 5th ed., chapter 3, Boston, USA, McGraw Hill.

PRINZ, W., 1999, "NESSIE: An Awareness Environment for Cooperative Settings". In: *Proceedings of the Sixth European Conference on Computer Supported Cooperative Work*, pp. 391-410, Copenhagen, Denmark.

RAMAKRISHNAN, R., GEHRKE, J., 2000, *Database Management Systems*. 2ª ed., McGraw-Hill.

RAMAMPIARO, H., NYGARD, M., 1999, "Cooperative Database System: A Constructive Review of Cooperative Transactions Models". In: *Proc. International Symposium on Database Applications in Non-Tradicional Environments (DANTE'99)*, pp. 315-324, IEEE Computer Society Press, Kyoto, Japan.

REDMILES, D. F., KANTOR, M. L., SOUZA, C. R. B., 2002, "Awareness Gauges". In: http://www.isr.uci.edu/projects/cassius/flier.pdf, Acessado em 11/2003.

ROSEMAN, M., GREENBERG, S., 1996, "Building Real Time Groupware with GroupKit: A Groupware ToolKit", *ACM Transactions on Computer Human Interactions*, v. 3, n. 1, pp. 66-106.

RUSSELL, C., 2002, "Java Data Objects (JDO) Specification - Final Release". In: http://jcp.org/aboutJava/communityprocess/final/jsr012/index.html, Acessado em 10/2003.

SAFFO, P., 2002, "Sensors: The Next Wave of Infotech Innovation. Institute for the Future". In: http://www.saffo.com/sensors.html, Acessado em 09/2003.

SOHLENKAMP, M., 1998, *Supporting Group Awareness in Multi-User Enviroment Through Perceptualization*, Dissertation M.Sc., Fachbereich Mathematik-Informatik der Universität - Gesamthochschule - Paderborn, http://orgwis.gmd.edu/projects/ POLITeam/poliawac/ms-diss/.

SOHLENKAMP, M., PRINZ, W., FUCHS, L., 2000, "POLIAwac: Design and Evaluation of an Awareness Enhanced Groupware Client", *AI & Society Journal*, v. 14, pp. 31-47.

SOLARMETRIC, 2003, "Kodo JDO". In: http://www.solarmetric.com/Software/Kodo_JDO/, Acessado em 10/2003.

SOUZA, R. P., COSTA, M. N., BRAGA, R. M. M., et al., 2002, "Software Components Retrieval Through Mediators and Web Search", *Journal of the Brazilian Computer Society*, v. 8, n. 2, pp. 55-63.

STAAB, S., BARNEKOW, T., 1999, "Towards Learning Notification Triggers". In: *Proc. International Workshop on Intelligent Workflow and Process Management*, Stockholm, Sweden.

SULAIMAN, A., 2002, *Mineração de Dados Endógenos*, Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

SUN, 2003a, "Java Abstract Windowing Toolkit (AWT) Event Model". In: http://java.sun.com/docs/books/tutorial/uiswing/overview/event.html, Acessado em 10/2003a.

SUN, 2003b, "Java Remote Method Invocation (RMI)". In: http://java.sun.com/products/jdk/rmi/, Acessado em 10/2003b.

SUN, 2003c, "JavaBeans Specification". In: http://java.sun.com/products/javabeans/docs/spec.html, Acessado em 10/2003c.

SUN, 2003d, "JDBC - Data Access API". In: http://java.sun.com/products/jdbc/, Acessado em 10/2003d.

SUZUKI, J., YAMAMOTO, Y., 1999, "Extending UML with Aspects: Aspect Support in the Design Phase". In: *Proc. 3rd Aspect Oriented Programming Workshop (ECOOP'99)*, v. Springer LNCS 1743, Lisbon, Portugal.

TAM, J., 2002, *Supporting Change Awareness in Visual Workspaces*, M.Sc., Department of Computer Science, University of Calgary, Alberta.

THOMSEN, E., 2002, *OLAP. Construindo sistemas de informações multidimensionais*. Tradução da 2ª ed. ed., Rio de Janeiro - RJ, Editora Campus.

TJDO, 2003, "TJDO". In: http://tjdo.sourceforge.net, Acessado em 06/2003.

TREVOR, J., RODDEN, T., BLAIR, G., 1993, "COLA: A lightweight activity model for CSCW". In: *Proc. European Computer Supported Cooperative Work (ECSCW'93)*, pp. 13-17, Milan.

TYAGI, S., VORBURGER, M., MCCAMMON, K., et al., 2003, *Core Java Data Objects*. 1st ed., Prentice Hall PTR / Sun Microsystems Press.

VERSANT, 2003, "Versant and Java Data Objects". In: http://www.versant.com/CF/developer/jdo/, Acessado em 10/2003.

VIEIRA, V., EL-JAICK, D., WERNER, C. M. L., et al., 2002, *Suporte de Bancos de Dados Convencionais a Aplicações Cooperativas*. Relatório Técnico 1/2002 - COPPE/UFRJ.

WEAR, A. W., GONG, Y., CHANG, K. H., 1995, "Database Management for Multimedia Distributed Collaborative Writing". In: *Proc. 33rd ACM Southeast Conference*, pp. 42-51, Clemson, SC.

WERNER, C. M. L., BORGES, M. R. S., MATTOSO, M. L. Q., et al., 2003, "OdysseyShare: Desenvolvimento Colaborativo de Componentes". In: *Anais Trilha de CSCW (WebMidia'2003) (a ser publicado)*, Salvador, Brasil.

WERNER, C. M. L., MANGAN, M. A. S., MURTA, L., 2002, "OdysseyShare: Um Ambiente para o Desenvolvimento Cooperativo de Componentes". In: *SBES - Caderno de Ferramentas*, pp. 444-449, Gramado, Brasil.

YAHOO, 2003, "YahooGroups Home Page". In: http://www.yahoogroups.com, Acessado em 01/3 A.D.

# ANEXO 1

## *ELEMENTOS DA PROGRAMAÇÃO ORIENTADA A ASPECTOS (AOP)*

Três elementos essenciais aparecem em todas as aplicações que desenvolvem AOP (Figura 31): classes, aspectos e *weaver*. As classes servem para encapsular os requisitos funcionais da aplicação, enquanto os aspectos encapsulam os requisitos transversais. O *weaver* executa o entrelaçamento de classes e aspectos para gerar o programa.

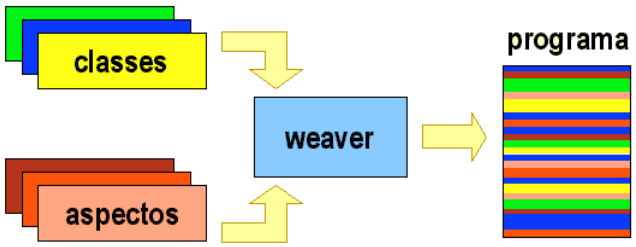



**Figura 31 – Construção de um programa usando AOP (PEREIRA, 2002)**

Dentre os benefícios esperados com a utilização de AOP, pode-se destacar:

- **Facilidade de desenvolvimento**: código-fonte menos entrelaçado, mais compacto e mais natural. Parte do desenvolvimento dos componentes são implementados separadamente, diminuindo o tamanho e complexidade desses componentes;
- **Facilidade de manutenção**: como os requisitos transversais não estão mais espalhados, o código-fonte da aplicação fica mais fácil de entender, depurar e modificar;
- **Facilidade de reutilização**: a AOP permite a utilização de hierarquia de aspectos, biblioteca de aspectos e aspectos "plug and play", ou seja, os aspectos podem ser conectados ou desconectados à aplicação de forma bastante trivial.

Apesar de ser uma tecnologia apontada como bastante promissora, a AOP apresenta, ainda, algumas deficiências, como por exemplo:

- **Tecnologia em evolução**: esse paradigma é, ainda, muito recente e, portanto, pouco testado na prática, em projetos reais;

- **Suporte a Análise e Projeto**: a modelagem de sistemas utilizando AOP ainda é um tópico em aberto. Algumas pesquisas recentes vêm trabalhando nessa linha (CHAVEZ e LUCENA, 2001).

## AspectJ

AspectJ é uma API que estende a linguagem Java com funcionalidades de AOP. Para os programadores familiarizados com Java, essa linguagem é bastante intuitiva e de fácil aprendizado, além de ser gratuita e ter seu código-fonte aberto e compartilhado. Fontes, executáveis e documentação podem ser adquiridos na página oficial do projeto (ASPECTJ TEAM, 2003). Um programa AspectJ pode ser projetado para ser revertido para um programa Java convencional, caso os aspectos sejam removidos. Desse modo, uma aplicação pode acoplar e desacoplar as funcionalidades implementadas pelos aspectos, de forma não muito complicada.

As principais diferenças entre aspectos e classes é que aspectos não possuem método construtor, logo não podem ser criados com o operador *new*, e incluem novos elementos em sua estrutura: *advices* e *pointcuts* que permitem a definição de *joinpoints*.

### *JoinPoint*

*Joinpoints* são pontos bem definidos na execução dinâmica de um programa (pontos de inserção), onde os aspectos podem interceptar as classes (KICZALES et al., 2001). Exemplos de *joinpoints* são (Figura 32): *Call* (chamadas de métodos, ponto em que um método é chamado), *Execution* (ponto em um método chamado quando a execução tem início), *Set* (ponto onde um valor é atribuído a um campo), *Get* (ponto onde um valor de um campo é obtido).

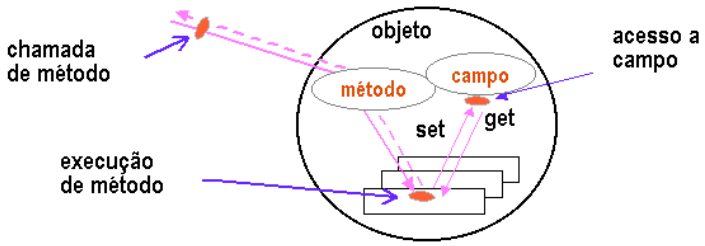



**Figura 32 – Exemplos de *JoinPoints* (PEREIRA, 2002)**

***Pointcut***

*Pointcuts* são predicados que definem coleções de *joinpoints* (KICZALES et al., 2001). *Joinpoints* não são executados individualmente, apenas como membros de *pointcuts*. Um *pointcut* particular assume um valor verdadeiro ou falso em qualquer *joinpoint* ao longo do código, ou seja, se o *joinpoint* corresponde ao predicado definido no *pointcut*, então este assume o valor verdadeiro, caso contrário, é falso.

A Figura 33 exibe dois *pointcuts,* nomeados *beginSession* e *endSession*, definidos no aspecto Sensor. Esses *pointcuts* descrevem, respectivamente, coleções de *joinpoints* em uma implementação JDO que indicam que uma conexão ou desconexão a um banco de dados foi efetuada.

```
//Intercepts the connection to the database
pointcut beginSession():
    execution(public * PersistenceManagerFactory.getPersistenceManager());

//Intercepts the connection being closed
pointcut endSession():
    execution(* PersistenceManager.close());
```

**Figura 33 – *Pointcuts* que descrevem ocorrência de eventos de sessão no JDO**

O *pointcut beginSession* indica que uma sessão teve início, ou seja, que a aplicação solicitou uma conexão ao banco de dados. Pela especificação JDO, esse evento ocorre quando o método *getPersistenceManager()* da interface *PersistenceManagerFactory* é executado. Isso é o que está descrito no código do *beginSession*. De forma similar, o *pointcut endSession* indica que a sessão foi encerrada e pela especificação JDO isso ocorre quando o método *close()* da interface *PersistenceManager* é executado.

***Advice***

*Advice* é um mecanismo utilizado para declarar que determinado código deve ser executado para cada um dos *joinpoints* definidos por um *pointcut* (KICZALES et al., 2001). A sintaxe de declaração de *advices* consiste do tipo, seguido pelo nome do *pointcut* e pelo corpo executável. Sempre que o controle de fluxo de um programa Java passa através de um *joinpoint* para o qual o *pointcut* é verdadeiro, o corpo do *advice* é executado. O AspectJ suporta três tipo de *advices*:

- *before*: o corpo do *advice* é executado imediatamente, antes de atingir o *joinpoint*;
- *after*: executado após atingir o *joinpoint*;
- *around*: o corpo do *advice* é executado no lugar do *joinpoint*, podendo ou não passar o controle a ele.

A Figura 34 exibe dois *advices* associados aos *pointcuts beginSession* e *endSession*. A variável *thisJoinPoint* é uma variável especial do AspectJ e tem função similar à variável *this* da programação Java convencional. Essa variável é visível apenas dentro do corpo de um *advice* e é instanciada a um objeto do tipo *JoinPoint* que encapsula o contexto do *joinpont* corrente. Seu objetivo é disponibilizar informações sobre o *joinpoint* que disparou o *advice*.

```
//Fires a Start Session Event
after(): beginSession() {
      fireSessionStarted(thisJoinPoint);

//Fires an End Session Event
after(): endSession() {
      fireSessionEnded(thisJoinPoint);
```

**Figura 34 – *Advices* que tratam eventos de sessão no Sensor**

```
protected void fireSessionStarted (JoinPoint jp) {
    PersistenceManagerFactory pmf = (PersistenceManagerFactory)jp.getThis();
    who = pmf.getConnectionUserName();
    where = new HashSet();
    where.add(pmf.getConnectionURL());
    how = jp.toLongString();
    why = null;
    awarenessServer = this.getServer();
    // What and When questions are filled in the Awareness Server
    // What means the event type represented by the method fired
    // When means current date/time
    awarenessServer.fireSessionStarted(who, where, how);
}
```

**Figura 35 –Coleta e distribuição das informações sobre a ação**

- Os métodos *fireSessionStarted* e *fireSessionEnded* são métodos do Sensor e recebem como parâmetro um objeto do tipo *JoinPoint*, correspondendo ao *joinpoint* que disparou o *advice*. Esse método obtém informações do *joinpoint* que permitem responder às questões 5W+1H referentes ao evento ocorrido (Figura 35).

# ANEXO 2

## *DEFINIÇÕES SOBRE ARMAZÉM DE DADOS E FERRAMENTAS OLAP*

Um armazém de dados (*Data warehouse*) é um banco de dados que contém dados que representam, geralmente, o histórico de negócios de uma organização, os quais são utilizados para análises que apóiem decisões de negócios tanto de planejamento estratégico quanto de avaliação de desempenho de uma unidade da organização (INMON, 1997).

*Data mart* é uma forma especial de *data warehouse*, normalmente contendo um subconjunto orientado a assunto dos dados da organização, apropriados para um grupo de negócios específico (KIMBALL e MERZ, 2000).

Processamento analítico (OLAP – *Online Analytical Processing*) refere-se ao conjunto de processos para criação, gerência e manipulação de dados multidimensionais para análise e visualização pelo usuário em busca de uma maior compreensão destes dados (CAMPOS, 2000). Apóia a tomada de decisões, permitindo analisar tendências e padrões em grandes quantidades de dados (derivados de sistemas operacionais) ao longo do tempo (histórico) e em diferentes localizações (geográfico) (CAMPOS, 2000).

A tecnologia OLAP permite que os *data warehouses* sejam utilizados de modo efetivo para consultas em tempo real, provendo respostas rápidas para consultas analíticas complexas (THOMSEN, 2002). O modelo de dados multidimensional e as técnicas de agregação presentes em ferramentas OLAP organizam e resumem grandes volumes de dados, permitindo que os mesmos possam ser analisados de forma rápida, utilizando ferramentas gráficas. As respostas a consultas sobre dados históricos geralmente levam a consultas subseqüentes à medida que o analista procura por respostas ou explora possibilidades. Os sistemas OLAP provêem a flexibilidade e rapidez para apoiar esse tipo de tarefa.

A Figura 36 mostra um esquema geral de geração de *data marts* a partir de dados operacionais e seu uso por ferramentas OLAP para apoiar analistas de negócio. O dado operacional deve passar por um processador ETL que executa a transformação dos dados para o modelo multidimensional, armazenando-os em um *datamart*. Esse

*datamart* é a base para geração do cubo que será analisado por analistas do negócio através de sistemas OLAP.

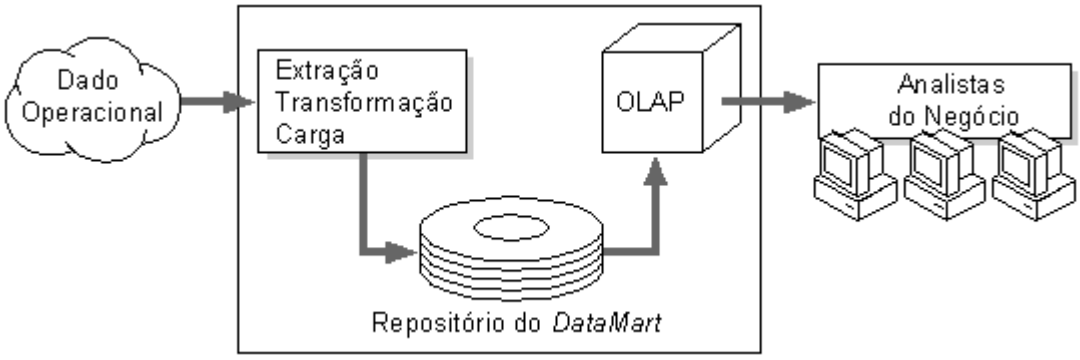



**Figura 36 – *Data mart* e OLAP (MICROSOFT, 2003a)**

Modelar os dados de forma multidimensional é uma forma de facilitar a análise de negócios em tempo real e aumentar o desempenho das consultas. Ferramentas OLAP permitem que dados armazenados em bancos de dados relacionais sejam transformados em informações de negócio com significado e fáceis de navegar, através da criação de cubos de dados. Uma forma bastante comum de modelar dados para uso multidimensional é através do esquema estrela. Esse esquema consiste de uma única tabela central, chamada tabela de fatos, conectada a diversas tabelas satélites, chamadas tabelas dimensões.

A tabela de fatos consiste dos dados numéricos que correspondem às medidas do cubo. As medidas de um cubo são valores quantitativos no banco de dados que se deseja analisar (CAMPOS, 2000). Medidas típicas são dados de vendas, custo e orçamento. Medidas são analisadas contra as diferentes categorias de dimensões do cubo. Por exemplo, pode-se desejar analisar dados de vendas e orçamento (as medidas) para um produto particular (dimensão) entre vários países (níveis específicos da dimensão local) durante dois anos específicos (níveis da dimensão tempo).

As dimensões de um cubo representam categorias distintas para analisar os dados do negócio, como por exemplo, tempo, local ou linha de produto (CAMPOS, 2000). As dimensões são geralmente organizadas em níveis hierárquicos. Cada nível pode ser agrupado para formar os valores para o próximo nível mais alto. Por exemplo, na dimensão tempo os dias podem ser agrupados em meses e os meses em semestres.

Hierarquias de dimensão são um dos principais instrumentos de análise de dados dimensionais, pois permitem que o analista aprofunde a observação das informações,

buscando informações mais detalhadas acerca de algum aspecto do negócio (CAMPOS, 2000).

Dentre as principais operações que podem ser executadas em sistemas OLAP estão (SULAIMAN, 2002) (CAMPOS, 2000):

- **Navegação pelas hierarquias e seus elementos**: permite selecionar as perspectivas sob as quais se deseja visualizar as variáveis;
- **Cruzamentos**: permitem sumarizar fatos segundo diferentes combinações das dimensões;
- ***Drill-down***: permite realizar navegação ao longo das dimensões na direção de maior detalhe (menor granularidade). Exemplo: visualizar as modificações realizadas em um determinado dia;
- ***Roll-up***: permite realizar navegação ao longo das dimensões na direção de menor detalhe (maior granularidade). Exemplo: visualizar as modificações realizadas em um determinado ano;
- **Rotação**: possibilita inverter colunas e linhas;
- ***Slice***: permite executar uma seleção em uma ou mais dimensões de um cubo de origem, resultando em um subcubo. Exemplo: eventos ocorridos sobre o artefato cuja classe se chama "modelo.Domínio" no dia "23/08/2003";
- **Cálculo e classificação**: permite executar cálculos sobre as células, classificando-as. Exemplo: as 5 classes que mais sofreram alteração, ou o usuário mais participativo (que mais executou ações sobre a base de dados).