3
Um Metamodelo para a Configuração de Espaços de Trabalho Virtuais Colaborativos

Neste capítulo é elaborado um metamodelo de multi-perspectiva para ajudar a definir e configurar a arquitetura do espaço de trabalho virtual colaborativo.

Os sistemas colaborativos são definidos como "uma combinação de tecnologia, pessoas e organizações que facilita a comunicação e a coordenação necessárias para que um grupo trabalhe efetivamente em conjunto em busca de um objetivo comum, obtendo ganhos para todos os membros." (Haynes et al., 2004) Empregando esse conceito, muitas empresas têm criado equipes virtuais de seus próprios funcionários, congregando profissionais geograficamente dispersos com capacidades complementares (Handel & Herbsleb, 2002; Bos et al., 2004), o que aumentou a demanda por aplicações CSCW. Para tornar mais fácil e efetivo o desenvolvimento de uma vasta gama dessas aplicações colaborativas, seria preciso oferecer uma arquitetura geral adaptável a diferentes situações, tarefas e configurações de uma forma flexível (Schuckmann et al., 1996).

Até o presente, a pesquisa em CSCW abordando as características da arquitetura mencionadas acima tem enfocado sobretudo as diferenças entre: *(i)* trabalho co-localizado e à distância (Bos et al., 2004; Dourish & Bly, 1992; Fussell et al., 2000; Herbsleb & Grinter, 1999; Herbsleb et al., 2000; Lauche, 2005); ou *(ii)* trabalho com pessoas da mesma cultura ou com bases comuns (afinidades, conhecimentos mútuos, crenças, objetivos, atitudes, etc) e trabalho com pessoas de diferentes culturas (trabalho inter-cultural) ou bases comuns (Fussell et al., 2000; Greenspan et al., 2000; Setlock et al., 2004). Essas perspectivas foram denominadas, respectivamente: *Centrada nos Lugares* e *Centrada nas Pessoas* (Jones et al., 2004).

Em vez de empregar uma dessas perspectivas para derivar um metamodelo para o projeto de aplicações CSCW, nossa proposta consiste em adotar uma visão diferente do problema, baseada nas atividades realizadas pelas equipes que

participam do trabalho colaborativo. Denominamos esta perspectiva *Centrada nas Atividades*. O termo *Atividade* aqui incorporado provém da teoria da atividade, sendo um conceito amplo e de múltiplos níveis.

Como detalharemos nas seções que se seguem, a perspectiva Centrada nas Atividades pode ser entendida como um conceito de multi-perspectiva (também de acordo com a forma como o termo foi incorporado a partir da teoria da atividade), visto que não somente engloba as perspectivas Centrada nos Lugares e Centrada nas Pessoas como também permite adotar cada uma delas, ou, tipicamente, ambas (de uma maneira híbrida) na medida desejada, além de admitir alterações passando de uma perspectiva para a outra.

É importante destacar que esta abordagem não se baseia na tecnologia, mas sim nos usuários ou nas pessoas (Ishii et al., 1994; Laurillau & Nigay, 2002; Palen, 1999; Sachs, 1995), enfocando um estudo qualitativo (Neale et al., 2004) das equipes envolvidas em vez de uma tecnologia específica.

A motivação para este trabalho foi a necessidade de configurar um espaço de trabalho virtual colaborativo para a gestão de desastres em estruturas de óleo e gás *offshore* de uma empresa global. Para identificarmos os requisitos do sistema, realizamos entrevistas semi-estruturadas com indivíduos chave, e não somente observamos sua prática profissional como efetivamente participamos de projetos e atividades conjuntas por mais de uma década.

3.1. Um Metamodelo Centrado nas Atividades

Iniciamos esta seção definindo *metamodelo*: trata-se de um modelo lógico – em contraposição a um modelo físico ou implementação – com ênfase na semântica declarativa em vez de nos detalhes de implementação do modelo. Espera-se que as implementações baseadas no metamodelo estejam de acordo com sua semântica (Athanasopoulos et al., 2003).

Diversos pesquisadores vêm desenvolvendo arquiteturas baseadas nesse conceito. A arquitetura colaborativa genérica de Dewan (1999) estrutura uma aplicação *groupware* em um número variável de camadas, desde o nível dependente do domínio até o nível de hardware, na qual uma camada é um componente de software que corresponde a um nível específico de abstração. De

forma semelhante, o metamodelo da arquitetura Clover (Laurillau & Nigay, 2002) também estrutura uma aplicação *groupware* em um número variável de camadas, decompondo cada camada em três sub-componentes funcionais dedicados à produção, comunicação e coordenação. No conjunto de ferramentas Prospero, Dourish propõe uma abordagem na qual uma arquitetura em metaníveis é usada pelos programadores para separar a representação e a funcionalidade de alto nível dos recursos de baixo nível, mapeando as primeiras nos segundos (Dourish, 1998).

O metamodelo que propomos adota uma abordagem semelhante de múltiplos níveis, de acordo com a versão da teoria da atividade de Leontjev (1978), na qual um esquema em três níveis descreve a estrutura hierárquica da atividade. Este esquema foi sintetizado por Tuikka (2002) da seguinte forma:

- O nível central é o das *ações*. As ações são orientadas visando *objetivos*. Em geral, os objetivos são subordinados a outros objetivos, os quais podem ser subordinados a outros, e assim por diante.
- Subindo na hierarquia de objetivos, finalmente chegamos a um objetivo no nível mais alto, que não está subordinado a nenhum outro. Este objetivo mais alto, que na teoria da atividade é denominado *motivo*, é o objeto de uma atividade completa.
- Descendo na hierarquia de ações, chegamos a processos automáticos que ajustam as ações às situações atuais. Segundo a teoria da atividade, tais processos são *operações*.
- As atividades, dirigidas pelos motivos, são realizadas por meio de ações que visam objetivos, os quais, por sua vez, são implementados por meio de operações.

De forma ortogonal a esta abordagem, semelhante ao metamodelo Clover, o metamodelo Centrado nas Atividades também permite desdobrar os componentes correspondes a um nível específico. Essas duas abordagens ortogonais aplicadas juntas contribuem para a generalidade de nosso metamodelo.

3.1.1.
Níveis de Abstração do Metamodelo

O nível mais alto de nosso metamodelo, o motivo, é representado por um *nó* complexo que engloba toda a atividade. O motivo pode ser muito variado, como a elaboração de um artigo (Figura 1) ou a gestão de um desastre em uma estrutura *offshore* de óleo e gás (Figura 2). O nível imediatamente abaixo do nível superior contém as principais ações que devem ser executadas para realizar toda a atividade. Essas ações resultam das interações dos grupos, com cada grupo sendo representado por um nó complexo e as interações entre eles sendo representadas por *arestas*, o que significa que cada nível de abstração pode ser representado por um grafo.

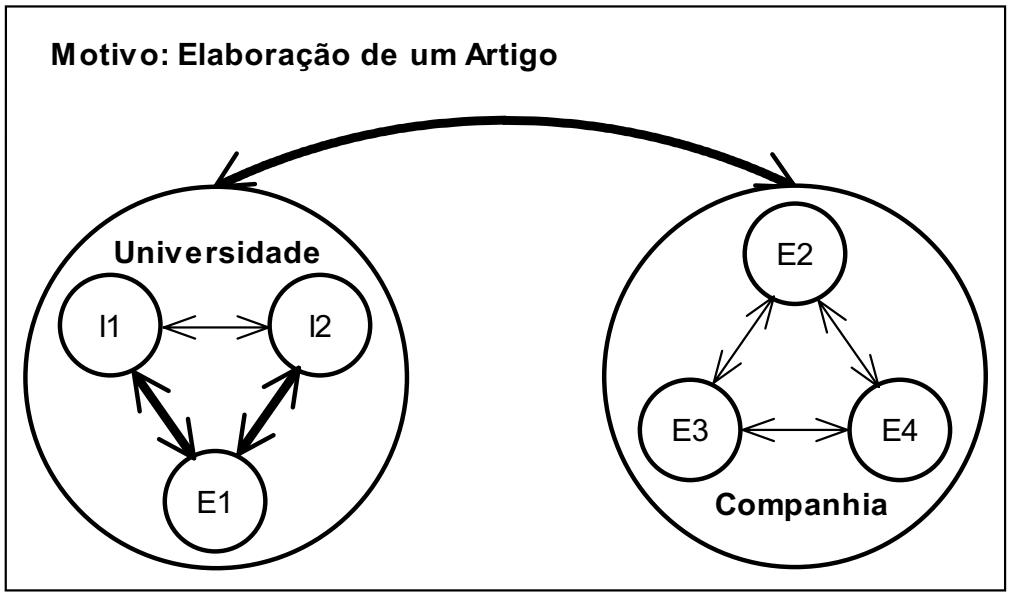

Figura 1 - Modelo colaborativo para a elaboração de um artigo: I1 e I2 são pesquisadores de Informática e E1, E2, E3 e E4 são Engenheiros

Continuamos descendo nos níveis, dividindo cada nó complexo do nível de abstração superior em nós mais elementares, até atingirmos um *nó folha*, que tipicamente será uma *pessoa*. A esses nós folhas associamos atributos de implementação e de hardware, tais como a aplicação a ser executada e o computador no qual ela deve rodar. Às vezes, o nó folha não é uma pessoa real mas um *agente de software*, responsável por um determinado conjunto de tarefas (Ellis et al., 1991).

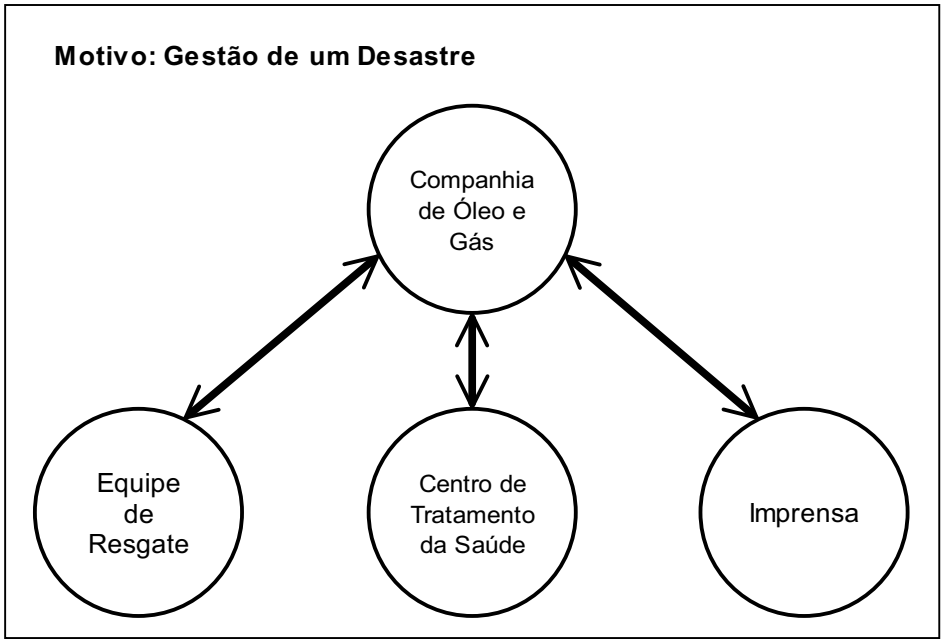

Figura 2 - Modelo colaborativo para a gestão de um desastre em uma estrutura *offshore* de óleo e gás

Na Figura 2, podemos ver que este metamodelo é suficientemente genérico para dar conta até mesmo de modelos que incluam grupos ou nós inter-organizacionais. Nesses casos, os mecanismos tradicionais de controle de acesso de *groupware* devem ser estendidos para lidar com aspectos novos, como privacidade, confidencialidade, interesses mútuos e contraditórios, culturas diferentes, etc (Cohen et al., 2000; Godefroid et al., 2000; Setlock et al., 2004; Stevens & Wulf, 2002).

3.1.2. Desdobrando os Componentes de um Nível de Abstração Específico

De forma ortogonal a esta abordagem, o metamodelo Centrado nas Atividades também permite desdobrar os componentes correspondentes a um nível específico. A idéia central desse mecanismo ficará mais clara se a ilustrarmos com um exemplo prático.

Consideremos um nível de abstração específico do exemplo do modelo de gestão de desastres: o primeiro nível abaixo do superior, relativo à companhia de óleo e gás (Figura 3). Observamos dois grupos principais: *Equipes Técnicas* e *Tomadores de Decisões.* Para facilitar a compressão do mecanismo de

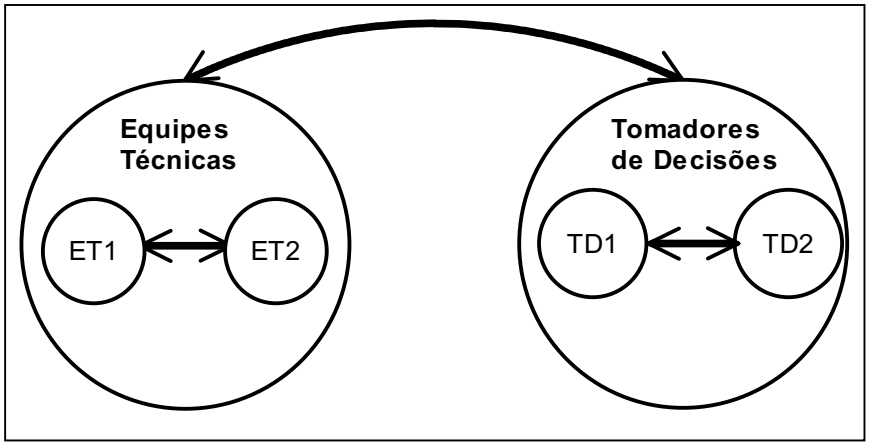

Figura 3 - Primeiro nível abaixo do superior, relativo à companhia de óleo e gás do modelo colaborativo de gestão de desastres: ET1 e ET2 representam a Equipe Técnica 1 e a Equipe Técnica 2, respectivamente, e TD1 e TD2 representam, respectivamente, o Tomador de Decisões 1 e o Tomador de Decisões 2

desdobramento, também representamos nessa figura o nível imediatamente abaixo do que estamos considerando: o grupo *Equipes Técnicas* é decomposto em dois subgrupos, *Equipe Técnica 1* e *Equipe Técnica 2*, e o grupo *Tomadores de Decisões* é decomposto em dois subgrupos (pessoas), *Tomador de Decisões 1* e *Tomador de Decisões 2*. Nessa configuração, ambos os *Tomadores de Decisões* estão sendo considerados como tendo os mesmos conhecimentos e o mesmo nível de interação com as *Equipes Técnicas.* Suponhamos agora que o *Tomador de Decisões 1* seja um gerente mais técnico e que o *Tomador de Decisões 2* seja um gerente de um nível organizacional mais alto dentro da empresa, como um diretor, e que não esteja tão envolvido tecnicamente com o problema. A forma mais simples de acomodar essa diferença é desdobrar o grupo *Tomadores de Decisões* em dois novos grupos nesse mesmo nível de abstração, *Tomador de Decisões 1* e *Tomador de Decisões 2*, cada um destes grupos com apenas um gerente. Devemos também substituir as arestas anteriores por novas, que representem as novas interações entre os novos grupos formados. A configuração resultante está mostrada na Figura 4. Agora está claro que o gerente no nível intermediário é o responsável pela comunicação direta com as *Equipes Técnicas* e pela comunicação com o diretor.

Este exemplo simples ilustra a utilidade deste mecanismo de desdobramento, que nos permite fazer experiências até chegar ao modelo mais apropriado para uma situação específica.

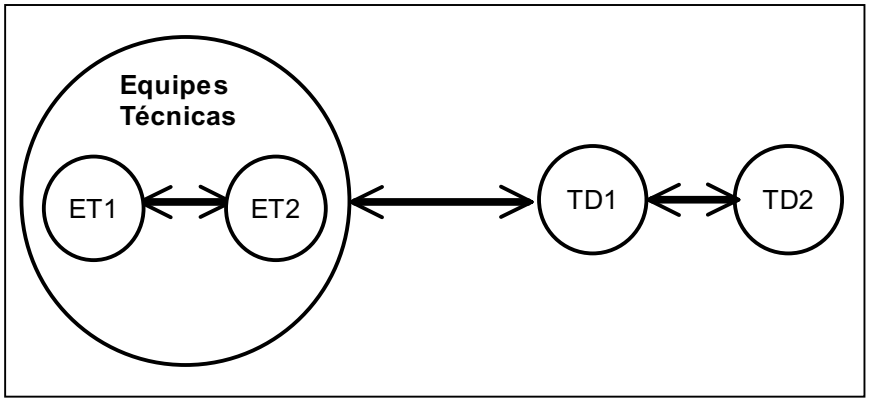

Figura 4 - Primeiro nível abaixo do superior, relativo à companhia de óleo e gás no modelo colaborativo de gestão de desastres, agora com o Tomador de Decisões 1 entre o nó das Equipes Técnicas e o Tomador de Decisões 2

3.1.3. Componentes do Metamodelo

Agora descreveremos os componentes principais de nosso metamodelo.

3.1.3.1. Nós

Os *nós* são componentes essenciais do metamodelo. Eles vão do nível mais alto, que representa toda a atividade (motivo), passando pelos muitos nós de diferentes níveis que representam grupos e subgrupos, até os *nós folhas*, que representam uma *pessoa* ou um *agente de software*. Os nós possuem um conjunto de *atributos*, como preferências de interface com o usuário e a língua utilizada, que são aplicados por meio de um conceito hierárquico de classe: o valor de um atributo de um nó vale para todos os sub-nós abaixo do nó considerado, a menos que seja explicitamente redefinido, sendo essa redefinição também válida para todos os níveis abaixo do nível considerado.

Os nós também possuem um atributo denominado *artefatos*, definido como "todos os objetos nos quais os usuários podem operar" (Gross & Prinz, 2004). Exemplos de artefatos são desenhos, rascunhos, modelos físicos, protótipos e descrições de produtos, documentos e arquivos, canetas e quadros negros, relatórios, artigos e comentários de revisão, planilhas e gráficos, todos em versão eletrônica, ou mais complexos, como calendários eletrônicos (Palen, 1999), pastas e armários compartilhados (Pankoke-Babatz & Syri, 1997), CAD e mapas

computadorizados (Pettersson et al., 2004), protótipos virtuais compartilhados (Tuikka, 2002), superfícies interativas compartilhadas (Brignull et al., 2004), monitores *tabletop* compartilhados (Morris et. al, 2004a), *tabletops* compartilhados entre múltiplos usuários (Morris et al., 2004b) e telas compartilhadas interativas (Paek et al., 2004). De acordo com o conceito de classe, um artefato associado ao nó de um grupo é compartilhado por todos os membros do grupo, a menos que seja explicitado de outra forma. Nesse caso, um mecanismo – tal como uma lista de controle de acesso – determinará quem possui acesso compartilhado ao artefato.

Um artefato possui duas *propriedades* básicas (Dourish & Bellotti, 1992): *sincronia*, que pode assumir os valores *síncrono* (há trabalho sendo realizado sobre ele neste momento) ou *assíncrono*; e *persistência*, que pode assumir os valores *persistente* (perdura após o trabalho sobre ele ter sido realizado) ou *volátil*.

Os nós folhas possuem atributos de nível baixo, que serão mapeados para uma implementação específica, tal como a aplicação a ser executada (Gross & Prinz, 2004).

3.1.3.2. Arestas

As *arestas* de nosso metamodelo representam os *caminhos de interação* entre os nós. Elas podem ser unidirecionadas ou bidirecionadas, representando os sentidos de interação possíveis. Quando uma aresta é representada por uma seta fina, isso significa que os nós em suas extremidades estão co-localizados. Quando a seta é grossa, ela indica que um nó está um local remoto em relação ao outro.

Uma aresta possui um ou mais *canais*, cada um representando o canal mediado eletronicamente que permite a comunicação entre dois nós (Greenspan et al., 2000).

Segundo Nardi et al. (2000), a comunicação é principalmente sobre o intercâmbio de informações. As informações trocadas podem ser de muitos tipos, como texto, gráfico, voz e vídeo. Fuks et al. (2005) listam alguns elementos que os canais de comunicação devem considerar para que o intercâmbio de informações ocorra: mídia (textual, falada, pictórica ou gestual), modo de transmissão (em blocos ou continuamente; síncrono ou assíncrono), políticas de

restrições, meta-informações (sobre a mensagem, como o assunto, data, prioridade e categoria), estrutura conversacional (linear, hierárquica ou em forma de rede) e caminhos conversacionais.

Um canal possui quatro propriedades básicas:

- *Sincronia* (Dourish & Bellotti, 1992; Greenspan et al., 2000): a distinção síncrono-assíncrono depende de se o conteúdo é comunicado ou escolhido. Entre os exemplos de canais síncronos ou em tempo real, incluem-se: mensagens instantâneas, *chat* e vídeo-conferência. Uma característica importante dessas aplicações síncronas é que elas transmitem informações de percepção de presença (*presence awareness*) (Dourish & Bellotti, 1992; Godefroid et al., 2000; Pankoke-Babatz & Syri, 1997) sobre os membros que participam da sessão de colaboração. Alguns exemplos de canais assíncronos são: correio eletrônico, conferência por computador estilo fórum e páginas da Web. Por um lado, os canais síncronos podem oferecer retorno em tempo real; por outro, os assíncronos oferecem maior controle sobre a criação de conteúdo e facilitam o arquivamento e o encaminhamento de mensagens. Além desses dois modos tradicionais, Dourish and Bellotti (1992) apresentam o termo *semi-síncrono*, que abrange os modos síncrono e assíncrono e é associado a espaços de trabalho compartilhados: os sistemas semi-síncronos apresentam informações sobre colaboradores co-presentes sincronamente, ao mesmo tempo em que representam atividades anteriores realizadas por outros colaboradores que não estejam sincronamente presentes.
- *Persistência* (Dourish & Bellotti, 1992; Pankoke-Babatz & Syri, 1997): o canal é denominado *volátil* se só pode oferecer informações no momento em que o evento ocorre, como no caso de vídeo-conferência. Por outro lado, é chamado *persistente* se pode oferecer informações após algum tempo de ausência, para informar sobre progressos intermediários, ou quando solicitado a dar mais detalhes, ou para informar sobre a história de um objeto. Um exemplo de canal persistente é um *chat* que tenha a capacidade de armazenar um

histórico da discussão (Handel & Herbsleb, 2002; Ribak et al., 2002) ou uma conferência por computador estilo fórum.

- *Simetria* (Greenspan et al., 2000): um canal é considerado *simétrico* se o receptor de uma mensagem puder responder com o mesmo tipo de mensagem. Um exemplo é uma ferramenta de e-mail que permita ao mesmo tempo ler, compor e enviar mensagens. A vídeo-telefonia é um canal totalmente *simétrico* que permite ao canal audiovisual transmitir nos dois sentidos, enquanto as transmissões por televisão e rádio são totalmente *assimétricas.* As aplicações de ensino à distância trazem a possibilidade de formas híbridas de comunicação, nas quais ambos os lados podem se ouvir mas só o apresentador é visível para os estudantes.
- *Mídia*: esta propriedade está relacionada à riqueza de mídia (Greenspan et al., 2000) do canal de comunicação, envolvendo, por exemplo, aspectos de som e imagem. A mídia da comunicação presencial não mediada por computador é considerada mais "rica" do que a audiovisual, a qual por sua vez é mais "rica" do que a comunicação somente por som ou somente por imagem.

3.2. Instanciação do Metamodelo Centrado nas Atividades: Modelos Centrados nas Atividades

Nesta seção, enfocaremos a característica mais importante de nosso metamodelo Centrado nas Atividades: a capacidade de acomodar múltiplas perspectivas, correspondentes a diferentes modelos, cada uma como uma instância do metamodelo.

Para facilitar a compreensão das múltiplas perspectivas, derivaremos modelos a partir do modelo colaborativo de elaboração de um artigo apresentado na Subseção 3.1.1, com a única diferença de que agora haverá pesquisadores em duas universidades diferentes.

3.2.1.
Um Modelo Centrado nos Lugares

Suponhamos que os aspectos mais importantes de nossa aplicação colaborativa estejam relacionados com o local onde as pessoas estão efetivamente trabalhando. Essa prioridade pode ter sido motivada, por exemplo, pela necessidade de empregar uma comunicação com mídia "rica", como a presencial (Greenspan et al., 2000), ou porque os diferentes lugares possuem instalações diferentes, como computadores *desktop* e salas de visualização, com comportamentos profissionais diferentes. Um possível modelo usando essa perspectiva Centrada nos Lugares para a aplicação colaborativa da elaboração do artigo é o mostrado na Figura 5.

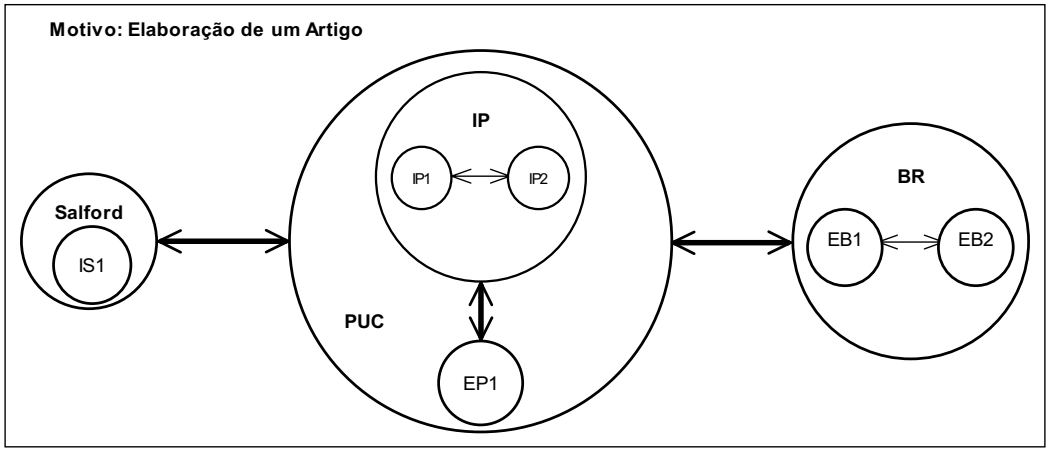

Figura 5 - Modelo colaborativo da elaboração de um artigo: perspectiva Centrada nos Lugares

Nessa figura, podemos observar que os grupos são formados a partir de uma perspectiva Centrada nos Lugares:

- Há três nós principais: a Universidade PUC, a Universidade de Salford e a Petrobras (BR). Visto que em nosso exemplo o artigo trata de Informática, o nó central, neste caso exercendo a função principal na preparação do artigo, é a PUC, que se comunica remotamente com a Universidade de Salford e com a Petrobras (neste modelo, não há vínculo direto entre Salford e a Petrobras).
- Dentro da PUC, há dois subgrupos: um é o Departamento de Informática (IP, I de Informática e P de PUC), que possui dois pesquisadores de Informática co-localizados e trabalhando em

conjunto, IP1 e IP2; e o outro é o Departamento de Engenharia, que possui somente um engenheiro (EP1). Os dois departamentos ficam em edifícios diferentes e também se comunicam, só que remotamente. Para não poluir visualmente a imagem, quando um nó contém somente um membro, esse nó será designado pelo nome do nó folha. É claro que é preciso ter cuidado com essa notação, pois pode ser interessante dispor de propriedades associadas a diferentes níveis, mesmo quando há somente um membro contido no nó pai, como no caso do nó EP1 do exemplo em questão. Esse pode ser o caso da hierarquia Universidade → Departamento → Pesquisador, com atributos específicos relativos a cada nível, mesmo com apenas um nó folha representando todos os grupos.

- Dentro da Universidade de Salford há apenas um pesquisador de Informática, chamado IS1 (I de Informática e S de Salford). Nesse caso, apenas para deixar clara a hierarquia de Salford, pusemos o nó folha IS1 dentro do nó pai Salford (conforme a notação acima, poderíamos apenas escrever IS1 dentro deste nó).
- Finalmente, dentro do nó Petrobras (BR), há dois engenheiros co-localizados trabalhando em conjunto: EB1 e EB2 (E de Engenheiro e B de BR).

Assim, nos modelos Centrados nos Lugares agrupamos os vários nós de modo a privilegiar os aspectos relacionados aos ambientes nos quais o trabalho está sendo realizado, como salas especiais (de visualização, por exemplo) ou dispositivos interativos especiais (como monitores interativos compartilhados), a necessidade de se ter comunicação presencial, etc.

3.2.2.
Um Modelo Centrado nas Pessoas

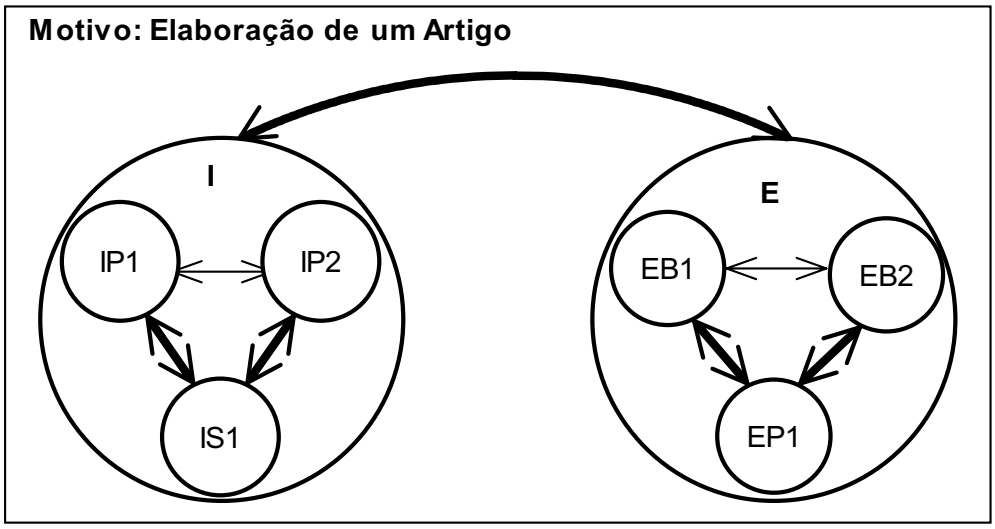

Figura 6 - Modelo colaborativo da elaboração de um artigo: perspectiva Centrada nas Pessoas

Consideremos agora que as questões principais de nossa aplicação colaborativa estão relacionadas às barreiras de cultura e bases comuns. Nesse caso, devemos derivar um modelo com uma perspectiva Centrada nas Pessoas, como o ilustrado na Figura 6:

- Agora há somente dois nós principais: o grupo de pesquisadores de Informática (I) e o grupo de Engenheiros (E), que se comunicam remotamente.
- Dentro do grupo I há três pesquisadores de Informática que trabalham juntos: IP1, IP2 e IS1. IP1 e IP2 estão co-localizados e IS1 fica localizado remotamente a ambos, possivelmente co-localizado em termos virtuais. É importante notar que, apesar de IS1 ser de outra universidade, quando comparado a IP1 e IP2, as bases comuns são tão intensas que ele pertence ao mesmo subgrupo que IP1 e IP2, apesar de estar localizado remotamente a eles. Se essa relação não for tão intensa, ou se houver uma barreira cultural, I deveria ser dividido em dois subgrupos, um com dois pesquisadores co-localizados, IP1 e IP2, que se comunicassem remotamente com IS1, único membro de outro subgrupo de Informática.

- O mesmo raciocínio pode ser aplicado ao grupo E. Há três Engenheiros, EP1, EB1 e EB2. EB1 e EB2 estão co-localizados e EP1 fica localizado remotamente a ambos, possivelmente co-localizado virtualmente. De forma análoga à análise sobre IS1, se houver algum tipo de barreira cultural entre EP1 e o grupo formado por EB1 e EB2, devemos dividir o grupo E em dois subgrupos, um com os Engenheiros EB1 e EB2 co-localizados comunicando-se remotamente com EP1, único membro de outro subgrupo de Engenharia.

Logo, nos modelos Centrados nas Pessoas, os principais aspectos levados em conta na definição dos grupos estão relacionados às características dos participantes: sua cultura ou bases comuns, suas especialidades, suas habilidades, comportamentos, etc.

3.2.3. Combinação de Perspectivas: um Modelo Centrado nas Atividades

Agora combinaremos as duas perspectivas anteriores na perspectiva que denominamos Centrada nas Atividades. No caso da aplicação colaborativa que estamos examinando, parece ser mais adequado enfocar a atividade completa que está sendo realizada – a elaboração de um artigo – e então derivar os grupos que serão formados. Para elaborar o artigo, os autores IP1, IP2, IS1 e EP1 procuram derivar um novo modelo teórico baseado nos requisitos identificados por meio de estudo de campo, trabalhando em conjunto com os Engenheiros EB1 e EB2. Assim, juntamos essas pessoas em dois grupos principais, formados com base em sua atividade principal: o grupo de Teoria e o grupo de Campo. Os grupos então são organizados de acordo com a Figura 7:

- Os dois grupos principais, Teoria e Campo, se comunicam remotamente entre si.
- Dentro do grupo Teoria há dois subgrupos: um com três pesquisadores de Informática (I), IP1 e IP2, trabalhando em conjunto e co-localizados, com IS1 trabalhando em posição remota, possivelmente co-localizado virtualmente; e outro com um só

Engenheiro, EP1, trabalhando em posição remota com relação ao grupo I.

- O grupo Campo é equivalente ao grupo BR na Figura 5, com dois Engenheiros, EB1 e EB2, co-localizados trabalhando em conjunto.

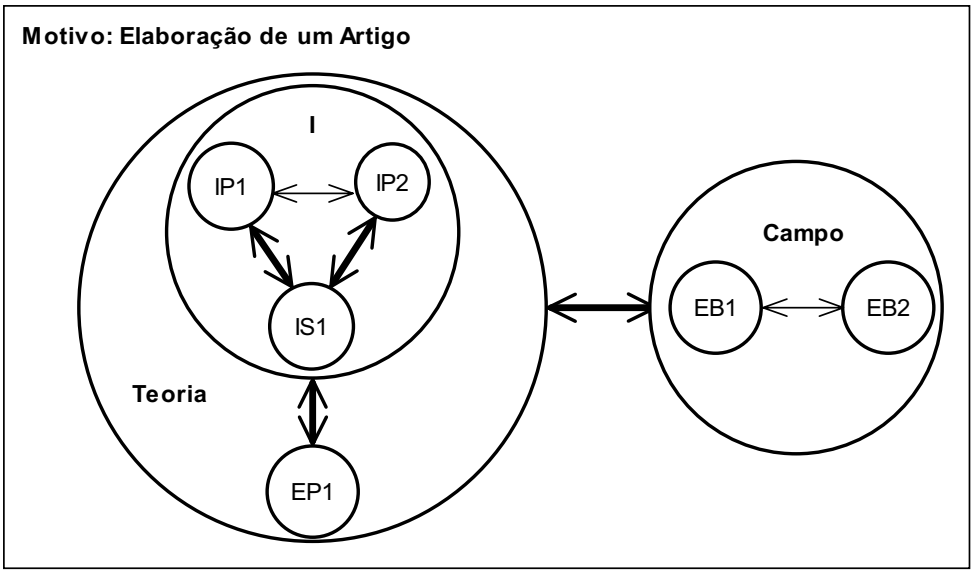

Figura 7 - Modelo colaborativo da elaboração de um artigo: perspectiva Centrada nas Atividades

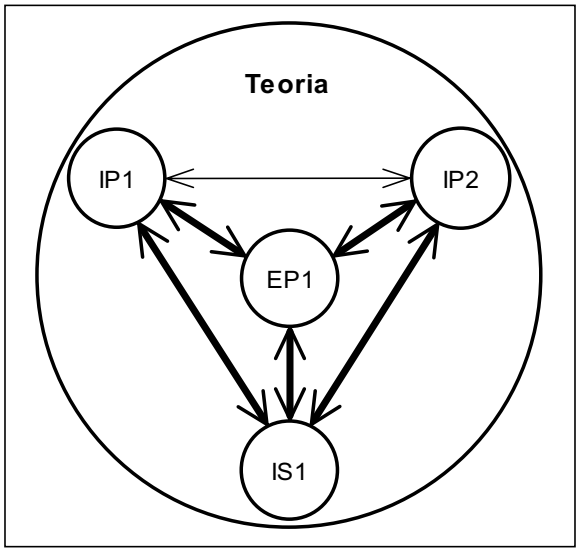

Figura 8 - Nova configuração do grupo Teoria

É importante reparar no papel desempenhado por EP1. Claramente, EP1 está sendo considerado de uma forma um pouco diferente dos outros membros do grupo Teoria, provavelmente agindo mais como um consultor para questões de

engenharia. Contudo, se considerássemos que EP1 tem as mesmas bases e cultura dos outros membros do grupo Teoria, capaz de ter com eles o mesmo nível de discussão, provavelmente seria melhor integrá-lo no mesmo grupo, ficando o novo grupo Teoria como o ilustrado na Figura 8.

3.2.4. Metamodelo Centrado nas Atividades: Algumas Conclusões

A partir dos exemplos acima, observamos a utilidade deste metamodelo Centrado nas Atividades, capaz de acomodar diferentes perspectivas e permitir ao projetista testar modelos e escolher aquele que melhor atende a seus requisitos. Observamos também que as necessidades pessoais e os recursos do espaço físico, tal como um espaço de trabalho compartilhado, orientam a configuração do modelo.

Outro aspecto importante que norteia a topologia do modelo é a informação que deve ficar disponível em cada nó definido para o modelo. Ao elaborar um modelo, é importante considerar não somente os aspectos topológicos, como também aqueles que envolvem a colaboração em si, ou seja, como os membros do grupo podem cooperar melhor por meio da produção, manipulação e organização de informações, além da construção e do aprimoramento de objetos de cooperação.

Também é importante saber reconfigurar o modelo, pois parece que algumas configurações funcionam melhor do que outras. Sem um reconfigurador automático, deve ser interessante ao menos derivar algumas configurações predefinidas adequadas, que pudessem ser selecionadas ao se iniciar uma aplicação colaborativa.

Deve-se destacar ainda que este metamodelo de multi-perspectiva não constitui, intrinsecamente, um metamodelo diferente quando comparado a muitos outros já desenvolvidos, de forma que podemos utilizar muitas linguagens de especificação disponíveis para descrevê-lo, como veremos mais adiante neste capítulo. O que parece ser a maior contribuição deste metamodelo é sua capacidade expressiva, que ajuda a esclarecer, por meio de uma representação gráfica simples, as relações entre os grupos envolvidos na atividade principal que está sendo realizada. Observamos com usuários reais que essa representação

simples lhes trouxe uma melhor compreensão do problema, além de estimulá-los a discutir e propor modificações na configuração dos modelos. As setas finas e grossas das arestas também transmitem rapidamente a noção de quem está localizado local ou remotamente em relação aos demais, além de indicar a necessidade de requisitos menos ou mais sofisticados em termos da rede de comunicação.

Por meio de uma análise detalhada, identificamos os três componentes do modelo 3C em nosso metamodelo Centrado nas Atividades, a saber: comunicação, coordenação e cooperação (Ellis et al., 1991; Fuks et al., 2005). O componente de cooperação às vezes recebe denominações diferentes, mas que representam os mesmos aspectos: colaboração (Ellis et al., 1991), produção (Laurillau & Nigay, 2002) e acoplamento de trabalho (Neale et al., 2004).

O componente de comunicação é o relacionado ao intercâmbio de mensagens e informações entre as pessoas (Fuks et al., 2005). Em nosso metamodelo, ele é representado pelas arestas que ligam os nós.

O componente de cooperação é a operação conjunta que ocorre durante uma sessão em um espaço compartilhado. Os membros de um grupo cooperam produzindo, manipulando e organizando informações, além de desenvolvendo e refinando artefatos. Podemos concluir que, em nosso metamodelo, o componente de cooperação é formado pela associação dos muitos níveis de abstração diferentes, que representam a atividade completa que está sendo realizada durante a sessão colaborativa, incluindo todas as aplicações que são executadas nos nós folhas e os artefatos que são produzidos ou manipulados – isto é, o espaço de trabalho que está sendo modelado. Todos esses elementos estão sendo armazenados durante a sessão colaborativa e podem ser utilizados como uma base de conhecimentos, seja para uma auditoria ou para uma reconfiguração do modelo em sessões futuras.

Finalmente, o componente de coordenação está relacionado ao gerenciamento de pessoas, suas atividades e recursos (Raposo & Fuks, 2002). Em uma atividade em grupo fracamente acoplada, na qual a seqüência de processamento é resultado das ações realizadas pelos participantes, a coordenação é feita implicitamente. Por outro lado, se o trabalho envolve atividades fortemente acopladas, há a necessidade de prescrever a ordem das ações, como ocorre em sistemas com fluxo de trabalho sistemático (Pankoke-Babatz & Syri, 1997).

Nosso metamodelo, com os elementos que foram considerados, é capaz de representar trabalho em grupo pouco acoplado. Para que o metamodelo também dê conta de atividades muito acopladas, identificamos a necessidade de introduzir novos elementos, que corresponderão à componente de coordenação do modelo 3C e serão descritos na próxima seção: *elementos de especialização das arestas, regras de papéis* e *tabela de atributos de mensagens*.

3.3. Metamodelo Centrado nas Atividades: Componentes de Coordenação

Conforme mencionado, identificamos a necessidade de introduzir novos elementos no metamodelo, correspondentes à componente de coordenação do modelo 3C, que descrevemos nas subseções que se seguem: *elementos de especialização das arestas, regras de papéis* e *tabela de atributos de mensagens*.

3.3.1. Elementos de Especialização das Arestas

Analisando uma mensagem através do caminho de interação desde um nó remetente até um nó receptor, identificamos alguns requisitos adicionais que nosso metamodelo deveria incluir. Suponhamos, por exemplo, uma situação em que há um grupo de engenheiros que utilizam computadores *desktop* e enviam um vídeo para outro grupo de engenheiros, localizados em uma sala de visualização, com uma tela de exibição maior. Seria interessante se o segundo grupo pudesse ajustar automaticamente algumas propriedades de vídeo para suas condições tecnológicas antes de exibi-lo na tela da sala de visualização. Isso poderia ser feito por meio do que denominamos *módulo de processamento pós-comunicação*, executado pelo nó receptor imediatamente após receber a mensagem através do canal de comunicação e imediatamente antes de encaminhá-la aos membros do grupo.

Outro exemplo é o uso de uma ferramenta de captura de vídeo durante uma sessão de simulação colaborativa. Dentro do grupo técnico, todos os membros devem receber todos os quadros de um simulador que é executado em um nó folha. Por outro lado, para os gerentes que fazem parte do grupo de tomadores de decisões, seria suficiente receber apenas um de cada dez quadros. Aqui, a solução

seria adicionar um *módulo de processamento pré-comunicação* a ser executado pelo nó remetente, como se fosse um filtro, imediatamente antes de enviar a mensagem (um quadro) através do canal de comunicação.

Um terceiro e mais abrangente exemplo seria o envio de uma mensagem de um grupo de brasileiros para um grupo de chineses, sendo o inglês a língua comum. No caso deste exemplo, consideremos que não há um tradutor português-chinês-português disponível. Neste caso, seria preciso empregar processamento pré e pós-comunicação. O primeiro seria executado pelo nó remetente, que traduziria a mensagem do português para o inglês imediatamente antes de enviá-la através do canal de comunicação. A mensagem trafegaria então pelo canal de comunicação e chegaria ao nó de destino. Lá, um módulo de processamento pós-comunicação traduziria a mensagem do inglês para o chinês, enviando-a aos receptores.

A partir dos exemplos acima, concluímos que seria interessante dividir esses módulos de processamento pré e pós-comunicação em duas classes diferentes:

- A primeira classe é constituída pelos módulos de pré e pós-processamento associados aos nós folhas. Eles representam os módulos de processamento a serem executados particularmente sobre uma mensagem específica enviada entre dois nós, os quais ficam armazenados em uma tabela especial com os identificadores (*id_mensagem*, *receptor*), como será descrito na Subseção 3.3.3.
- A segunda classe é a constituída pelos módulos de pré e pós-processamento associados a nós grupos em diferentes níveis da hierarquia do metamodelo, representando as políticas desses grupos, respectivamente, ao enviar (políticas de envio) e receber (políticas de recebimento) mensagens. Visto que não há políticas particulares relacionadas a uma mensagem específica, mas sim políticas mais gerais associadas a grupos em diferentes níveis do metamodelo, elas podem ser armazenadas no próprio metamodelo.

Para ilustrar melhor esta idéia, apresentamos na Figura 9 os diferentes módulos de processamento pré e pós-comunicação que podem ser executados quando uma mensagem é enviada de um Pesquisador de Informática PI1, do

Departamento de Informática DI1 da Universidade U1, para um Pesquisador de Informática PI2, do Departamento de Informática DI2 da Universidade U2.

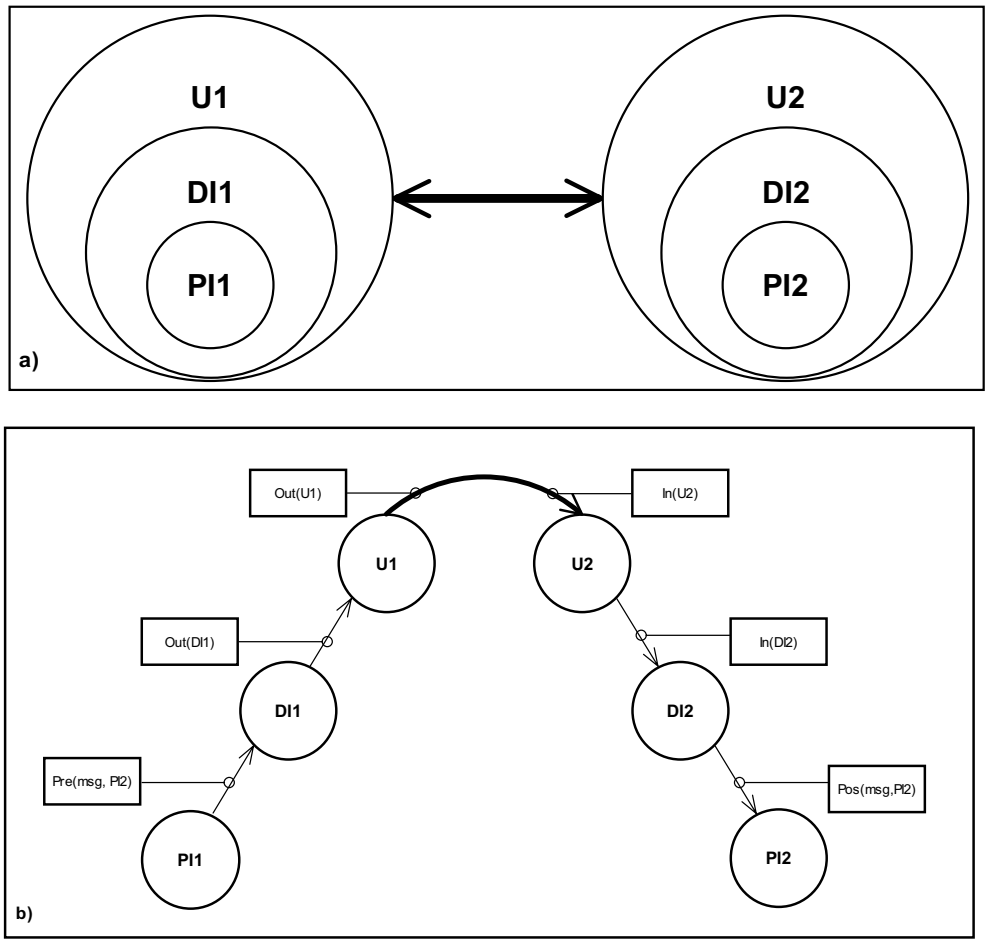

Figura 9 - Metamodelo Centrado nas Atividades, processamento pré e pós-comunicação: a) ponto-de-vista do metamodelo; b) ponto-de-vista do pré e pós-processamento

Na Figura 9 é possível acompanhar a seqüência dos módulos de processamento executados:

- O primeiro módulo de pré-processamento a ser executado é o associado com a mensagem em particular que está sendo enviada de PI1 para PI2.
- Então, as políticas de envio associadas ao departamento DI1 são executadas.
- Finalmente, as políticas de envio associadas à universidade U1 são executadas.
- Então a mensagem é enviada através da aresta de comunicação específica (que liga os nós superiores de cada lado).

- Agora, do lado receptor da aresta, o primeiro módulo de pós-processamento a ser executado são as políticas de recebimento associadas à universidade U2.
- Então, as políticas de recebimento associadas ao departamento DI2 são executadas.
- Finalmente, é executado o terceiro módulo de pós-processamento, associado com a mensagem em particular que está sendo enviada de PI1 para PI2.

Analisando este exemplo, surgem duas questões importantes. A primeira diz respeito a quem executará esses módulos de processamento pré e pós-comunicação. Em relação à aresta, do lado do remetente, o candidato natural para executar os módulos de pré-processamento é o nó folha que está enviando a mensagem. Do lado do receptor, isso poderia ser feito adicionando-se um atributo ao primeiro nó para o qual a aresta aponta (no exemplo, a universidade U2), que corresponde ao nó folha desse grupo que executará os módulos de pós-processamento.

A segunda questão envolve os algoritmos a serem executados quando uma mensagem é enviada, tanto do lado do remetente quanto do receptor. Apresentaremos esses algoritmos mais adiante, após definir os outros dois componentes de coordenação de nosso metamodelo, nas próximas subseções: as regras de papéis e a tabela de atributos de mensagens.

Os elementos de especialização das arestas aumentam a flexibilidade do metamodelo em dois aspectos. Um é a concentração de valores de atributos nos módulos que podem ser modificados dinamicamente em tempo de execução. O segundo aspecto enfatiza a adequação do modelo aos protocolos sociais (Morris, 2004b) envolvendo aplicações colaborativas. Como exemplo desses protocolos sociais, em nosso metamodelo o processamento pós-comunicação transfere para uma folha de um nó grupo receptor a responsabilidade de decidir a quais membros deste grupo a mensagem recebida será encaminhada. Isso parece razoável, visto que ela conhece bem o grupo e foi por ele designado para assumir esse papel. Outro exemplo: retomando o exemplo prático já visto, sobre a ferramenta de captura de vídeo, o grupo remetente poderia considerar mais educado e conveniente enviar todos os quadros do vídeo para o grupo de tomadores de

decisões e deixá-lo decidir se deseja cortar alguns quadros ou não (o filtro seria incluído no processamento de pós-comunicação). É claro que essa segunda abordagem aumentaria a carga da rede, mas esse seria o preço de um protocolo social mais "diplomático".

Outros exemplos práticos de uso de pré e pós-processamento:

- Conversão de formatos de arquivos (ex.: CAD) de um sistema de um lado da aresta para outro sistema do outro lado. Isto poderia ser feito em qualquer um ou em ambos os lados se, por exemplo, um arquivo neutro fosse transmitido através do canal de comunicação.
- Transformação de coordenadas UTM para XYZ (em qualquer um dos lados).
- Procedimentos de detecção de vírus e políticas de *firewall* (em qualquer um dos lados).

Para finalizar esta subseção, é interessante destacar que esses conceitos de pré e pós-processamento são geralmente levados em conta em estudos sobre aplicações colaborativas, embora nem sempre seguindo a mesma abordagem. Por exemplo, empregando a mesma abordagem, Cortés e Mishra (1996) utilizam funções de *pré* e *pós-execução* em seus programas de coordenação DCWPL. David e Borges (2001) também propõem o uso de filtros de entrada e saída pela aplicação e pelo usuário para assegurar a privacidade das informações, selecionar os eventos, informações de percepção (*awareness*) e/ou para modificar o processamento desses eventos. Dourish (1998) usa esses mesmos conceitos no conjunto de ferramentas Prospero, denominando-os *métodos anteriores* e *métodos posteriores*, os quais são executados pelo conjunto de ferramentas. Finalmente, Stevens e Wulf (2002), empregando uma abordagem diferente, utilizam os termos *ex-ante*, *uno-tempore* e *ex-post*, associados ao momento em que as permissões de controle de acesso são definidas.

3.3.2. Regras de Papéis

Os estudos em CSCW têm empregado *regras de papéis* para a estrutura de coordenação por mais de uma década. Por exemplo, Furuta e Stotts (1994)

apresentam uma evolução do modelo Trellis na qual os protocolos de interação entre os grupos são representados de forma separada dos processos de interface que utilizam esses protocolos para a coordenação. Os protocolos são interpretados, de modo que as interações entre os grupos podem ser modificadas à medida que a tarefa em colaboração progride. As alterações podem ser feitas ou por uma pessoa, editando as especificações do protocolo em tempo real, ou por um processo de "observação silenciosa". O Trellis combina navegação em hipermídia com suporte à colaboração – um *hiperprograma* – com esse modelo baseado sobretudo em uma rede de Petri temporizada por transição, executada sincronamente, que é a estrutura do hiperprograma. A notação de rede utilizada é denominada CTN (*colored timed nets* – redes coloridas com indicação de tempo). Um hiperprograma Trellis é completado por meio da inserção de anotações sobre os componentes da CTN: a CTN é a descrição da tarefa e as diferentes anotações são as informações necessárias para realizar a tarefa. Uma categoria de anotação importante é o *conteúdo*. Fragmentos de informação (textos, gráficos, vídeos, áudio, códigos executáveis, outros hiperprogramas) são associados a lugares na CTN. Outra categoria de anotação são os *eventos*, que são mapeados nas transições da CTN. Uma terceira categoria de anotação é o par *atributo/valor (A/V)*. Uma lista de pares A/V é mantida em cada local, cada transição, cada arco e para a CTN como um todo.

Edwards (1996) apresenta o Intermezzo, sistema que permite aos usuários controlar a colaboração por meio de *políticas* que funcionam como regras gerais para restringir e definir o comportamento do sistema, em reação ao estado do mundo. As políticas são descritas em termos dos direitos de controle de acesso aos objetos de dados, e são associadas a grupos de usuários denominados *papéis.* Os papéis representam não somente conjuntos de usuários definidos estaticamente, como também descrições dinâmicas de usuários que são avaliadas enquanto as aplicações são executadas. Esse aspecto em tempo de execução dos papéis permite que eles reajam de forma flexível ao dinamismo inerente à colaboração. O Intermezzo oferece mecanismos para criar papéis estáticos e dinâmicos, uma implementação de políticas sobre o sistema de controle de acesso "bruto" e uma linguagem de especificação declarativa para os papéis e as políticas que pode ser utilizada para controlar e configurar o subsistema de políticas. Enquanto os papéis estáticos são implementados por meio de listas simples de controle de acesso, os

papéis dinâmicos são implementados associando-se uma função de predicado a um conjunto de direitos de controle de acesso.

Para Cortés e Mishra (1996), um programa colaborativo deve ser dividido em dois componentes principais: um programa computacional que modele os artefatos compartilháveis e um programa de coordenação que especifique a forma como esses artefatos devem ser compartilhados. Os programas de coordenação podem ser facilmente modificados, muitas vezes sem nenhuma alteração no programa computacional. Os autores desenvolveram uma linguagem de coordenação – DCWPL (*Describing Collaborative Work Programming Language* – Linguagem de Programação para a Descrição de Trabalho Colaborativo – pronunciada "*decouple*") – e seu intérprete em tempo de execução para que os programadores criem programas de coordenação, especificando mecanismos de controle separados dos programas computacionais, que são escritos em uma linguagem de programação tradicional. Um programa de coordenação em DCWPL é composto de uma ou mais primitivas de coordenação: artefatos, papéis, armazenamento, funções de coordenação, políticas e gerenciamento de sessão.

Li e Muntz (1998) propõem a COCA (*Collaborative Object Coordination Architecture* – Arquitetura de Coordenação de Objetos Colaborativos) para ser um *framework* genérico para o desenvolvimento de sistemas colaborativos evolutivos e a modelagem de políticas de coordenação. Assim como nos três trabalhos já mencionados, eles também são a favor de separar coordenação e computação. COCA oferece aos desenvolvedores uma linguagem de especificação baseada em lógica para modelar as políticas de coordenação. Nessa arquitetura, os participantes de uma colaboração são divididos por papéis e são definidas regras ativas para cada papel, de modo a resguardar suas interações com os outros colaboradores. Essas políticas são interpretadas em tempo de execução pela máquina virtual de COCA, da qual uma cópia é executada na máquina de cada participante. Essa abordagem torna conveniente fazer alterações tanto durante o desenvolvimento como em tempo de execução. As políticas de COCA incluem não somente controle de acesso como também controle de sessão, controle de concorrência, tomada de vez, etc.

Roussev et al. (2000) adotam uma abordagem baseada em componentes para separar dados e controle. O estado compartilhado de um componente de dados é modelado como propriedades do componente, e as alterações sobre seu

estado são modeladas como eventos de alteração das propriedades. O projeto componível baseia-se em padrões de programação que eliminam a ligação implícita entre a estrutura lógica de um objeto compartilhado e abstrações particulares definidas pelo sistema, o que aumenta a gama de objetos suportados e permite a extensibilidade.

Laurillau e Nigay (2002) apresentam o modelo arquitetônico Clover, resultante da combinação da abordagem em camadas da arquitetura genérica de Dewan (1999) com a decomposição funcional do modelo Clover. O modelo Clover define três classes de serviços que uma aplicação *groupware* pode suportar: serviços de produção, comunicação e coordenação. Essas três classes de serviços podem ser encontradas em cada camada funcional do modelo. A partição funcional de Clover estabelece um mapeamento direto entre os conceitos de projeto e a modelagem da arquitetura de software. Essa partição funciona como um guia na organização das funcionalidades identificadas durante a fase de projeto. Além disso, ela complementa a partição tradicional das funcionalidades em componentes, na qual cada um corresponde a um nível de abstração – por exemplo, na arquitetura de Dewan. De fato, para um componente correspondente a um nível de abstração, o metamodelo Clover defende três sub-componentes dedicados à produção, comunicação e coordenação. Essa implementação modular está de acordo com as propriedades de modificação, reutilização e extensibilidade. No metamodelo Clover, as funcionalidades de coordenação, produção e comunicação são todas tratadas da mesma forma, diferentemente de outros estudos, como a arquitetura de Dewan, na qual as funcionalidades de coordenação são tratadas de forma especial, mas não as de produção e comunicação. A arquitetura conceitual é mapeada para uma arquitetura de implementação designando processos para os componentes; os processos, por sua vez, são atribuídos aos computadores. Podem ser aplicadas diferentes abordagens de distribuição a uma arquitetura conceitual. A distribuição e a decomposição de camadas (ou decomposição de componentes de software) são dois mecanismos ortogonais. Em particular, uma camada compartilhada no nível conceitual não implica um ponto central no nível da implementação.

Então, Yang e Li (2004) retomam a abordagem baseada em componentes empregada previamente por Roussev et al. (2000), também separando dados e controle, mas agora suportando protocolos adaptáveis de consistência em sistemas

colaborativos. Ao separar claramente dados e controle, eles permitem que os protocolos de consistência sejam vinculados dinamicamente aos dados compartilhados no nível do objeto. Os protocolos podem ser comutados em tempo de execução sem modificar o código fonte.

Em consonância com todos esses estudos, exceto de algum modo pelo modelo Clover, decidimos adotar a estratégia de separar a estrutura de coordenação e o programa computacional. Em nosso metamodelo, utilizamos uma linguagem de especificação baseada em lógica para especificar as políticas de coordenação. Forneceremos mais detalhes sobre essa linguagem de especificação no Capítulo 5. Por ora, destacamos os seguintes pontos:

- Declaramos uma barra de colaboração, que é usada para conectar todos os participantes desta colaboração; a barra de colaboração deve ter pelo menos uma declaração de canal. Permitimos a execução de muitas colaborações diferentes ao mesmo tempo, cada uma com sua barra de colaboração correspondente.
- Os papéis são definidos especificando-se comportamentos e restrições individuais em diferentes conjuntos de regras lógicas.
- A comunicação entre os participantes ocorre através de um ou mais canais de mensagens associados a uma barra de colaboração.
- As tarefas básicas de recebimento e envio de mensagens são realizadas por:
    - Para receber mensagens, utilizamos uma regra ativa chamada *on-arrive*, com os argumentos *canal*, *receptor*, *id_mensagem* (e *remetente*).
    - Para enviar mensagens, usamos uma fórmula *send* com os argumentos *canal*, *remetente, id_mensagem* (e *receptor*).

Explicamos por que os últimos argumentos dessas tarefas estão entre parênteses:

- Para a tarefa de receber mensagens, *remetente* é opcional. Para a tarefa de enviar mensagens, se por exemplo for adotado *multicast* de IP (*Internet Protocol* – Protocolo de Internet) como o modelo de comunicação do grupo, a mensagem é enviada a todos os receptores,

tornando o argumento *receptor* desnecessário (no caso de um serviço *unicast*, ele seria necessário).

- No entanto, o principal motivo para pôr esses argumentos entre parênteses é que, para aumentar a flexibilidade, decidimos construir uma *tabela de atributos de mensagens* que inclui o *remetente*, o *receptor* e outros atributos de cada mensagem, como veremos na próxima subseção. Nesse caso, a alteração do receptor de uma mensagem seria feita apenas alterando-se uma linha da tabela, sem modificar o programa de coordenação. Fizemos também algumas modificações na regra e na fórmula usada na COCA (Li & Muntz, 1998), as quais se transformaram respectivamente em:
  - *on-arrive*(canal,tab_mensagem(*dummy*,id_mensagem,receptor));
  - *send*(canal,tab_mensagem(remetente,id_mensagem,*dummy*)).

Um típico programa de coordenação de nosso metamodelo teria um formato semelhante ao apresentado na Tabela 1. Nele, *self* é um *functor* que denota o participante atual, *source* associa a identificação do participante com a mensagem e *display* é empregado para exibir o conteúdo da mensagem na tela.

Acompanhando esse programa de coordenação linha a linha, temos:

- A primeira linha definindo a colaboração, com o nome *simulacao_integrada*.
- A segunda linha definindo uma barra de colaboração com apenas um canal *remoto*.
- A terceira linha abrindo um conjunto de regras correspondentes ao papel *tecnico_0* (que pode ser instanciado pelos participantes durante a execução da colaboração).
- E então, definimos as três regras que constituem o papel *tecnico_0*:
  - A primeira regra, *on-init*, é disparada quando a colaboração *simulacao_integrada* é iniciada. Quando isso ocorre, uma mensagem com *id_mensagem* 1 é enviada ao receptor determinado pela linha na tabela de atributos de mensagens com chaves *remetente*=*source*(*self*) (i.e., o participante que está instanciando o papel de *tecnico_0*) e *id_mensagem*=1.

- A segunda regra, *on-arrive*, é ativada quando uma mensagem com *id_mensagem* 2 é recebida por *source*(*self*). Quando isso ocorre, a mensagem é exibida no console do participante.
- Finalmente, a terceira regra, outra *on-arrive*, é ativada quando uma mensagem com *id_mensagem* 3 é recebida por *source*(*self*). Quando isso ocorre, a mensagem é exibida no console do participante.

```
collaboration simulacao_integrada
{ collaboration_bus { channel(remoto). }
  role tecnico_0 //técnico responsável pela coordenação das simulações
  {
        on-init(simulacao_integrada) :-
              send(remoto, tab_mensagem(source(self), 1, dummy)).
        on-arrive(remoto, tab_mensagem(dummy, 2, source(self))) :-
              display(tab_mensagem(dummy, 2, source(self))).
        on-arrive(remoto, tab_mensagem(dummy, 3, source(self))) :-
              display(tab_mensagem(dummy, 3, source(self))).
  }
}
```

Tabela 1 - Metamodelo Centrado nas Atividades: programa de coordenação típico

De modo geral, as regras de papéis definem políticas de coordenação para cada papel presente em uma colaboração. Quando essas regras também determinam a ordem na qual os eventos ocorrem, elas também definem as regras do fluxo de trabalho.

3.3.3. Tabela de Atributos de Mensagens

Foram apresentados os módulos de processamento pré e pós-comunicação associados a cada mensagem enviada, então a escolha natural é montar uma tabela na qual possamos definir quais módulos de processamento devem ser executados a cada momento. Isso também facilita a alteração do nome de um módulo de pré ou pós-processamento em particular, mesmo em tempo de execução (além da possibilidade de alterar também o código em tempo de execução antes de ele ser invocado).

Assim, elaboramos uma *tabela de atributos de mensagens* em nosso metamodelo visando aumentar a flexibilidade do programa de coordenação, separando as regras de coordenação dos dados especificamente relacionados com cada mensagem, em consonância com o conceito geral do metamodelo de separar dados e controle, com este acesso indireto permitindo a reconfiguração dinâmica. A única exceção a esse conceito é que decidimos incluir o receptor (relativo ao controle, não aos dados) como uma coluna da tabela. Isso foi motivado pelo fato de que, em fluxos de trabalho fortemente acoplados, parece interessante ao menos alterar os receptores de uma dada mensagem em tempo de execução, sem a necessidade de modificar o programa de coordenação. Por exemplo, uma pessoa do grupo remetente pode não saber exatamente para quem a mensagem deve ser enviada, do lado dos receptores. Então, ela pode enviar a mensagem para o nó abstrato do grupo receptor e permitir que este determine, por meio do módulo de pós-processamento, a quais receptores a mensagem deve ser encaminhada. Outro exemplo seria o de um observador (p.ex., um gerente não responsável por tomar as decisões) que entrasse depois na sessão colaborativa e desejasse receber todas as mensagens a partir daquele momento. Seria uma questão de editar a tabela, incluindo novas linhas que associassem as mensagens futuras ao novo participante. Repare que as arestas que conectam esse novo participante aos demais devem estar previstas e incluídas no modelo antes do início da colaboração, e que o novo participante deve assumir um papel previamente definido que não interfira no processo principal.

A Tabela 2 apresenta as primeiras linhas de uma tabela de atributos de mensagens típica associada a uma aplicação colaborativa fortemente acoplada.

remetente	id_mensagem	receptor	aresta	pré-processamento	pós-processamento
T0	1	T1	a1	Pre_1_T1	Pos_1_T1
T1	2	T0	a1	Pre_2_T0	Pos_2_T0
T1	2	G1	a2	Pre_2_G1	Pos_2_G1
T1	3	T0	a1	Pre_3_T0	Pos_3_T0
T1	3	G1	a2	Pre_3_G1	Pos_3_G1

Tabela 2 - Colunas e primeiras linhas de uma tabela de atributos de mensagens típica

As chaves desta tabela são *remetente*, *id_mensagem* e *receptor*. Conforme já visto na Subseção 3.3.2, entramos na tabela utilizando o par (*remetente*, *id_mensagem*) ao enviar uma mensagem e utilizamos o par (*id_mensagem*, *receptor*) para receber uma mensagem.

3.3.4. Algoritmos de Remetente e Receptor

Apresentamos agora os algoritmos executados quando uma mensagem é enviada, tanto do lado do remetente quanto do receptor.

Observamos primeiro que cada mensagem tem um remetente inicial, que é necessariamente um nó folha, e um ou mais receptores finais, que podem ser folhas ou grupos. Já tendo apresentado as regras de papéis e a tabela de atributos de mensagens, descrevemos agora os algoritmos a serem executados de cada um dos lados, como mostrado na Tabela 3.

Consideremos a tabela de atributos de mensagens apresentada na subseção anterior (Tabela 2) para analisar os algoritmos e estabelecer em que momentos os módulos de pré e pós-processamento são executados para cada mensagem. É importante enfatizar duas questões:

- G1 (Grupo 1) não é um nó folha, mas um nó grupo. De acordo com os algoritmos, o módulo de pós-processamento *Pos_2_G1* é responsável por determinar a quais nós folhas do grupo G1 a mensagem (2, G1) deve ser encaminhada.
- Se por um lado enviar a mesma mensagem para cada receptor confere flexibilidade a nosso metamodelo, com a possibilidade de executar módulos de processamento pré e pós-comunicação particulares para cada receptor, além de várias vantagens, como as mencionadas nos exemplos fornecidos na Subseção 3.3.1, por outro é preciso considerar o impacto dessa estratégia na performance. Conforme relatam Li e Muntz (1998), os primeiros modelos eram fortemente baseados no agrupamento de múltiplas comunicações ponto-a-ponto. Nesse caso, um pacote com *n* receptores pretendidos deve ser enviado pelo remetente *n* vezes, independentemente da localização física das partes envolvidas. Essa abordagem produz custo extra tanto para os remetentes da mensagem quanto para os canais de comunicação, e a latência aumenta com o tamanho da população de receptores. Portanto, é preciso equilibrar nossas necessidades ao escolher a estratégia a ser usada na aplicação

colaborativa. Por exemplo, nosso algoritmo parece ser adequado se houver poucos nós e muita diversidade entre os nós. Por outro lado, poderia ser mais eficiente escolher um modelo de comunicação com *multicast* de IP se houver muitos nós com não muita diversidade entre eles.

sender (*remetente*, *receptor*, *flag*)
- *until* o *receptor* ser encontrado *repeat*
  - no nível atual, procura a sub-árvore que contém o *receptor*
  - *if* o *receptor* for encontrado (e todo o caminho do *remetente* até o *receptor* for determinado)
    - *if flag = na_tabela*
      - executa o módulo de pré-processamento associado com o par (*mensagem*, *receptor*)
    - *else*
      - cria uma nova linha na tabela de atributos de mensagens com o par (*mensagem*, *receptor*), indicando o módulo de pós-processamento a ser executado
    - executa todas as políticas de envio associadas aos grupos nos níveis do caminho que começa no *remetente* até que seja atingida a aresta de comunicação
    - envia a mensagem com o *receptor* para o nó folha designado no atributo de pós-processamento do nó grupo *receptor*, ou para o próprio *receptor*
  - *else*
    - vai para o nível superior

Lado do remetente da aresta de comunicação (executado pelo nó folha *remetente_inicial*):
- *for* cada *receptor_final* associado com a mensagem
  - *sender* (*remetente_inicial*, *receptor_final*, *na_tabela*)

Lado do receptor da aresta de comunicação:
- recebe a mensagem
- executa todas as políticas de recebimento associadas aos grupos nos níveis do caminho que começa no nó atual até que seja atingido o nó *receptor_final*
- *if receptor_final* é um nó folha
  - *if receptor_final* é igual ao nó de execução de pós-processamento
    - executa o módulo de pós-processamento associado com o par (*mensagem*, *receptor_final*)
  - *else*
    - envia a mensagem para *receptor_final*
- *else*
  - executa o módulo de pós-processamento associado com o par (*mensagem*, *receptor_final*), que neste caso deve determinar o(s) nó(s) folha ou grupo(s) que receberá(ão) a mensagem
  - *for* cada nó determinado acima (*no_corrente*)
    - *sender* (*receptor_final*, *no_corrente*, *nao_na_tabela*)

Tabela 3 - Metamodelo Centrado nas Atividades: algoritmos de remetente e receptor

Descrevemos agora os dois algoritmos, do lado do remetente e do lado do receptor, apresentados na Tabela 3. Primeiro detalharemos um método comum, chamado *sender*, que é utilizado dos dois lados. O objetivo básico deste método é encontrar, no componente de rede de nosso metamodelo, o nó receptor que receberá a mensagem, determinando o caminho completo do remetente até o receptor. O método tem três argumentos: *remetente*, *receptor* e uma *flag* que determina se a mensagem atual está ou não na tabela de atributos de mensagens. Ele possui um laço principal *until...repeat* para encontrar o *receptor*, que é executado começando pela busca do *receptor* na sub-árvore definida pelo nível imediatamente acima do nível do *remetente* (um nó folha), indo para cima

executando o mesmo procedimento até que se encontre o *receptor*. Quando isso ocorre, há duas situações possíveis:

- A mensagem já está na tabela de atributos de mensagens:
  neste caso, o módulo de pré-processamento associado com o par (*id_mensagem*, *receptor*) é executado.
- A mensagem não está na tabela de atributos de mensagens:
  isso significa que se trata de uma nova mensagem que está sendo criada em tempo de execução por um nó grupo do lado do receptor, definindo uma nova linha na tabela que inclui o módulo de pós-processamento.

Finalizando o método *sender*, há outros dois passos:

- O primeiro executa todas as políticas de envio associadas aos grupos nos níveis do caminho que começa no *remetente* até que seja atingida a aresta de comunicação.
- O segundo finalmente envia a mensagem com o *receptor* para o nó folha designado no atributo de pós-processamento do nó grupo *receptor*, ou para o próprio *receptor* (se for um nó folha).

Descrevemos agora o algoritmo do lado do remetente. Com o método *sender* definido acima, este algoritmo, executado pelo nó folha *remetente_inicial*, é simplesmente um laço *for* que repete o passo a seguir para cada *receptor_final* (determinado nas linhas da tabela de atributos de mensagens) associado com a mensagem atual que está sendo enviada:

- *sender* (*remetente_inicial*, *receptor_final*, *na_tabela*).

Podemos ver que o método é invocado com *flag=na_tabela*, o que significa que essas mensagens já estão predefinidas na tabela de atributos de mensagens.

Finalmente, descrevemos agora o algoritmo do lado do receptor:

- O primeiro passo, executado pelo nó designado no atributo de pós-processamento do nó grupo *receptor* ou pelo próprio *receptor* (se for um nó folha), consiste exatamente em receber a mensagem.

- O segundo passo executa todas as políticas de recebimento associadas aos grupos nos níveis do caminho que começa no nó atual até que seja atingido o nó *receptor_final*.
- O terceiro e último passo principal é executado de forma diferente dependendo de se o *receptor_final* é ou não é um nó folha:
  - Se o *receptor_final* é um nó folha:
    - O *receptor_final* pode ser o próprio nó de execução do pós-processamento, quando simplesmente executa o módulo de pós-processamento associado com o par (*id_mensagem*, *receptor_final*), ou pode não ser, caso em que o nó de execução do pós-processamento ainda precisa enviar a mensagem ao nó folha *receptor_final*.
  - Se o *receptor_final* é um nó grupo, são executados dois passos:
    - O módulo de pós-processamento associado com o par (*id_mensagem*, *receptor_final*) é executado, caso em que deve determinar o(s) nó(s) folha(s) ou grupo(s) para receber a mensagem.
    - Um laço *for* que repete o seguinte passo para cada nó determinado no passo acima (o *no_corrente*) é executado:
      - *sender* (*receptor_final*,*no_corrente*,*nao_na_tabela*).

      Podemos ver que, neste caso, o método *sender* é invocado com *flag=nao_na_tabela*, o que indica que se tratam de mensagens novas que estão sendo criadas em tempo de execução pelo nó grupo.

3.4. Metamodelo Centrado nas Atividades: Linguagem de Especificação para o Componente de Rede

Como foi visto na Seção 3.2, o componente principal de nosso metamodelo Centrado nas Atividades é a rede constituída por nós e arestas que definem, respectivamente, os grupos que realizam a atividade e os caminhos da interação entre eles.

É preciso utilizar uma linguagem de especificação para descrever a rede, escolhendo dentre muitas linguagens de especificação disponíveis, tais como *Process Algebra* (Milner, 1989), *Petri Nets* (Murata, 1989), *Unified Modeling Language* (UML) *Diagrams* (UML, 2006) e *Use Case Maps* (UCM) (Buhr & Casselman, 1996). Escolhemos a linguagem de especificação proposta (Russo, 1988; Santos, 1986) para o projeto EITIS (*Environment of Integrated Tools for Interactive Systems* – Ambiente de Ferramentas Integradas para Sistemas Interativos), da PUC-Rio (Melo, 1987), principalmente por duas razões:

- Sua simplicidade, inspirada em uma notação BNF (Backus-Naur-Form) modificada, que nos permite manter o foco no trabalho que está sendo desenvolvido.
- O fato de que já a havíamos utilizado em um projeto anterior do Departamento de Informática da PUC-Rio.

A seguir, descrevemos brevemente a notação utilizada em nossos blocos de especificação:

- Em caixa alta, representamos as entidades e os relacionamentos, assim como os atributos que estão sendo descritos e os tipos e funções das entidades.
- Em caixa baixa, representamos os atributos que não estão sendo descritos e a sintaxe associada aos atributos que estão sendo descritos.
- O sinal + indica uma possível ocorrência de mais de uma instância de um objeto específico.
- O sinal | representa um "OU" lógico.
- Os relacionamentos ou atributos entre colchetes podem ou não ocorrer em uma instância específica da entidade que está sendo considerada.

Os blocos de especificação que definem as entidades e os relacionamentos principais do componente de rede de nosso metamodelo são mostrados na Tabela 4.

NÓ: id-nó
DESCRIÇÃO: texto
NOME: texto
TIPO: FOLHA | GRUPO
[PAI: id-nó]
[NÓ DE EXECUÇÃO DE PÓS-PROCESSAMENTO: id-nó]
[COMPUTADOR: id-computador]
[ATRIBUTOS: {nome-atrib tipo-atrib valor-atrib}+] //preferências interface c/usuário, linguagem
[COMPARTILHA: id-artefato+]
[POLÍTICA DE RECEBIMENTO: id-política]
[POLÍTICA DE ENVIO: id-política]

ARESTA: id-aresta
DESCRIÇÃO: texto
DISTÂNCIA: LOCAL | REMOTA
DIREÇÃO: UNIDIRECIONADA | BIDIRECIONADA
REMETENTE: id-nó
RECEPTOR: id-nó
POSSUI: id-canal+

ARTEFATO: id-artefato
DESCRIÇÃO: texto
NOME: texto
TIPO: vários
SINCRONIA: SÍNCRONO | ASSÍNCRONO
PERSISTÊNCIA: PERSISTENTE | VOLÁTIL
[LCA: id-lca] //Lista de Controle de Acesso

CANAL: id-canal
DESCRIÇÃO: texto
NOME: texto
TIPO: vários
SINCRONIA: SÍNCRONO | ASSÍNCRONO
PERSISTÊNCIA: PERSISTENTE | VOLÁTIL

Tabela 4 - Componente de rede: blocos de especificação que definem entidades e relacionamentos

Então, esses blocos de especificação devem ser armazenados em uma estrutura de dados com uma DDL (*Data Description Language* – Linguagem de Descrição de Dados) semelhante à mostrada na Tabela 5 (onde estão descritas somente as entidades principais do componente de rede, i.e., nós e arestas).

```
RECORD no
ITEM id-no NUM 2 KEY
ITEM descricao CHAR 35
ITEM nome CHAR 30
ITEM tipo CHAR 5
ITEM pai NUM 2
ITEM pos NUM 2
ITEM computador NUM 12
.
.

RECORD aresta
ITEM id-aresta NUM 4 KEY
ITEM descricao CHAR 35
ITEM distancia CHAR 6
ITEM direcao CHAR 11
ITEM remetente NUM 2
ITEM receptor NUM 2
.
.
```

Tabela 5 - Componente de rede: Linguagem de Descrição de Dados (DDL)

Finalmente, apresentamos na Tabela 6 os registros de carga para os nós e arestas correspondentes à Linguagem de Descrição de Dados da Tabela 5.

Registros de Carga

Nó

tipo-reg	id-no	descricao	nome	tipo	pai	pos	computador	…

Aresta

tipo-reg	id-aresta	descricao	distancia	direcao	remetente	receptor	…

Tabela 6 - Componente de rede: registros de carga para nós e arestas

3.5.
Modelo Centrado nas Atividades: Um Exemplo Simples Completo

Consideremos agora um exemplo simples, porém completo (Figura 10) para ilustrar como descrevemos os três componentes principais de nosso metamodelo: a rede, as regras de papéis e a tabela de atributos de mensagens.

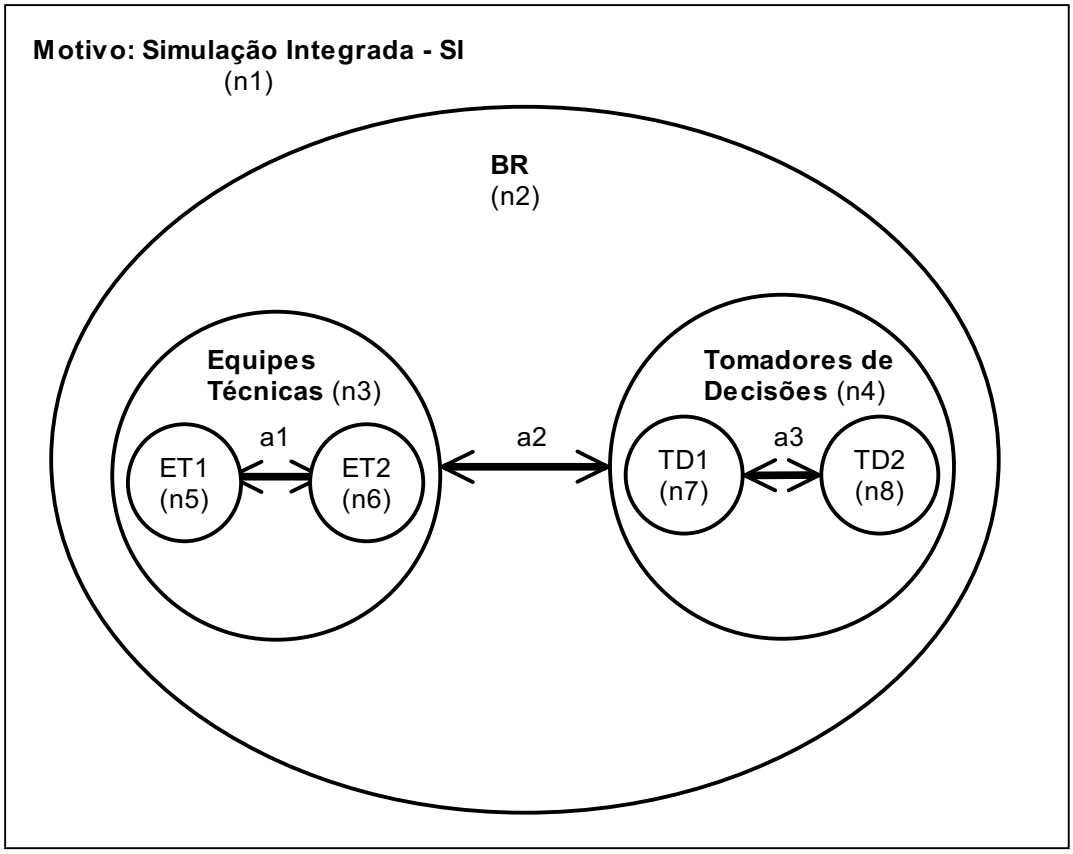

Figura 10 - Um exemplo simples completo de um modelo centrado nas atividades: Simulação Integrada

O nó do nível superior deste exemplo, o motivo, é a Simulação Integrada (n1). Ele tem um nó principal, a companhia de óleo e gás BR (n2), constituída por dois grupos: Equipes Técnicas (n3) e Tomadores de Decisões (n4).

Equipes Técnicas é formado por dois nós folhas, o técnico ET1 (n5) e o técnico ET2 (n6); Tomadores de Decisões é formado por dois outros nós folhas, o tomador de decisões TD1 (n7) e o tomador de decisões TD2 (n8).

Também representadas no componente de rede estão as arestas a1, a2 e a3, que mostram os caminhos da interação entre os nós, todas com setas grossas (remotas).

Os registros de carga desses nós e arestas estão apresentados na Tabela 7.

1 | 1 | Simulação Integrada | SI | GRUPO | 0 | 5 | …
1 | 2 | Companhia de Óleo e Gás | BR | GRUPO | 1 | 5 | …
1 | 3 | Equipes Técnicas | ET | GRUPO | 2 | 5 | …
1 | 4 | Tomadores de Decisões | TD | GRUPO | 2 | 7 | …
1 | 5 | Técnico 1 | ET1 | FOLHA | 3 | 0 | …
1 | 6 | Técnico 2 | ET2 | FOLHA | 3 | 0 | …
1 | 7 | Tomador de Decisões 1 | TD1 | FOLHA | 4 | 0 | …
1 | 8 | Tomador de Decisões 2 | TD2 | FOLHA | 4 | 0 | …
2 | 1 | canal de técnicos | REMOTA | BIDIRECIONADA | 5 | 6 | …
2 | 2 | canal técnico-gerencial | REMOTA | BIDIRECIONADA | 3 | 4 | …
2 | 3 | canal de gerentes | REMOTA | BIDIRECIONADA | 7 | 8 | …

Tabela 7 - Modelo Centrado nas Atividades: registros de carga do componente de rede

Nesta tabela, podemos identificar os oito nós e as três arestas que constituem o componente de rede do modelo de simulação integrada. Em termos dos nós, vemos que os registros definem quais nós são grupos e quais são folhas, quais são os pais dos nós (0, quando o pai é o motivo) e quais são os nós de execução de pós-processamento dos grupos (0, quando os nós são folhas). Por exemplo, a terceira linha (1 | 3 | Equipes Técnicas | ET | GRUPO | 2 | 5 | …) começa definindo que ela se refere a um nó (registro tipo 1), seguido pela sua identificação (3), descrição (Equipes Técnicas) e nome (ET). Então, o tipo de nó é definido como GRUPO e seu pai é definido como o nó com *id*=2 (definido na linha anterior – BR). Finalmente, o último atributo apresentado define o nó que executa o módulo de pós-processamento associado a este grupo (neste caso, o nó com *id*=5 definido duas linhas abaixo – ET1).

Quanto às arestas, identificamos quais são as remotas (neste exemplo, todas) e as locais, quais são as bidirecionadas (novamente, todas as deste exemplo) e as unidirecionadas, e os nós que definem as arestas. Por exemplo, a última linha (2 | 3 | canal de gerentes | REMOTA | BIDIRECIONADA | 7 | 8 | …) começa definindo que ela se refere a uma aresta (registro tipo 2), seguida de sua *id* (3) e descrição (canal de gerentes). Então, a aresta é definida como REMOTA e BIDIRECIONADA. Finalmente, os dois últimos atributos identificam os nós conectados pela aresta (neste caso, os nós com *id* 7 e 8 – TD1 e TD2, respectivamente).

A Tabela 8 apresenta as regras de papéis definidas para *tecnico_1* instanciadas por ET1 do grupo Equipes Técnicas.

```
collaboration simulacao_integrada
{ collaboration_bus { channel(remoto). }
  role tecnico_1 //Técnico 1 de Equipes Técnicas, responsável pela coordenação das simulações
  {
  on-init(simulacao_integrada) :-
    send(remoto, tab_mensagem(source(self), 1, dummy)). //envia "Técnico 2, por favor inicie a simulação."
  on-arrive(remoto, tab_mensagem(dummy, 2, source(self))) :-
    display(tab_mensagem(dummy, 2, source(self))). //exibe "A simulação foi iniciada."
  on-arrive(remoto, tab_mensagem(dummy, 3, source(self))) :-
    display(tab_mensagem(dummy, 3, source(self))), //exibe "Fim da simulação."
    send(remoto, tab_mensagem(source(self), 4, dummy)). //envia "Tomador de Decisões, por favor valide a
                                                        simulação."
  on-arrive(remoto, tab_mensagem(dummy, 5, source(self))) :-
    display(tab_mensagem(dummy, 5, source(self))). //exibe "Simulação validada."
  }
}
```

Tabela 8 - Modelo Centrado nas Atividades: regras de papéis para *tecnico_1*

Acompanhando esse programa de coordenação linha a linha, temos:

- A primeira linha definindo a colaboração, com o nome *simulacao_integrada*.
- A segunda linha definindo uma barra de colaboração com apenas um canal *remoto*.
- A terceira linha abrindo um conjunto de regras correspondentes ao papel *tecnico_1* (que, neste exemplo, é instanciado pelo participante ET1).
- Então, definimos as quatro regras que constituem o papel *tecnico_1*:
  - A primeira regra, *on-init*, é disparada quando a colaboração *simulacao_integrada* é iniciada. Quando isso ocorre, uma mensagem com *id_mensagem* 1 e conteúdo "Técnico 2, por favor inicie a simulação." é enviada ao receptor determinado pela linha na tabela de atributos de mensagens com a chave *sender=source*(*self*) (i.e., ET1) e *id_mensagem*=1. Na Tabela 9, essa linha é a primeira e indica que o *receptor* é ET2.
  - A segunda regra, *on-arrive*, é ativada quando uma mensagem com *id_mensagem* 2 é recebida por *source*(*self*) (i.e., ET1).

Quando isso ocorre, a mensagem é exibida no console do participante: “A simulação foi iniciada.”

- A terceira regra, outra *on-arrive*, é ativada quando uma mensagem com *id_mensagem* 3 é recebida por *source*(*self*) (i.e., ET1). Quando isso ocorre, a mensagem 3 é exibida no console do participante: “Fim da simulação.” Além disso, uma mensagem com *id_mensagem* 4 e o conteúdo “Tomador de Decisões, por favor valide a simulação.” é enviada ao receptor determinado pela linha na tabela de atributos de mensagens com a chave *sender*=*source*(*self*) (i.e., ET1) e *id_mensagem*=4. Na Tabela 9, esta linha é a sexta e indica que o *receptor* é TD.
- Finalmente, a quarta regra, mais uma *on-arrive*, é ativada quando uma mensagem com *id_mensagem* 5 é recebida por *source*(*self*) (i.e., ET1). Quando isso ocorre, a mensagem é exibida no console do participante: “Simulação validada.”

De forma semelhante, poderíamos definir regras de papéis para outros participantes da aplicação colaborativa. Vale notar que, uma vez que neste exemplo as regras de papéis definem a ordem dos eventos, elas também podem ser consideradas regras para fluxo de trabalho.

Finalizamos nosso exemplo simples apresentando a tabela de atributos de mensagens (Tabela 9) com todas as mensagens enviadas pelos participantes, a qual, juntamente com as regras de papéis, define a estrutura de coordenação de nosso metamodelo.

remetente	id_mensagem	receptor	aresta	pré-processamento	pós-processamento
ET1	1	ET2	a1	Pre_1_ET2	Pos_1_ET2
ET2	2	ET1	a1	Pre_2_ET1	Pos_2_ET1
ET2	2	TD	a2	Pre_2_TD	Pos_2_TD
ET2	3	ET1	a1	Pre_3_ET1	Pos_3_ET1
ET2	3	TD	a2	Pre_3_TD	Pos_3_TD
ET1	4	TD	a2	Pre_4_TD	Pos_4_TD
TD1	5	ET1	a2	Pre_5_ET1	Pos_5_ET1

Tabela 9 - Metamodelo Centrado nas Atividades: tabela de atributos de mensagens para o modelo BR

Nesta tabela, vemos também os diferentes módulos de pré e pós-processamento a serem executados ao enviar as mensagens, em particular as mensagens 2 e 3, que têm diferentes módulos de processamento associados a diferentes receptores. Também é importante observar que a mensagem 2 enviada por ET2, a mensagem 3 enviada por ET2 e a mensagem 4 enviada por ET1 destinam-se ao grupo TD. Isso significa que os módulos de pós-processamento *Pos_2_TD*, *Pos_3_TD* e *Pos_4_TD*, respectivamente, todos executados pelo nó de execução de pós-processamento do grupo TD, TD1 (veja a Tabela 7, na qual ele está com *id* 7), devem determinar quais nós folhas do grupo TD devem receber as mensagens. Neste exemplo, definimos que as mensagens 2 e 3 são re-encaminhadas para TD1 e TD2 (visto que são apenas mensagens de acompanhamento), e que a mensagem 4 é encaminhada a TD1, já que ele é o responsável por tomar a decisão final sobre a simulação integrada.

Finalizamos esta seção observando que é preciso garantir a consistência entre as três partes do metamodelo. Isso significa, por exemplo, que os tipos de canais presentes nas regras de papéis, local ou remoto, devem ser os mesmos representados no componente de rede (setas finas ou grossas). Por sua vez, as mensagens presentes na tabela de atributos de mensagens somente devem ser definidas se as arestas correspondentes, ligando os remetentes e os receptores no componente de rede, existirem. Os algoritmos do remetente e do receptor devem verificar essa consistência em tempo de execução. Finalmente, observamos que, nas fórmulas *send* utilizadas nas regras de papéis, visto que os receptores são determinados apenas indiretamente por meio da tabela de atributos de mensagens, indicamos apenas o canal usado para a comunicação com o receptor principal da mensagem. Os canais utilizados na comunicação com os outros receptores são determinados ao se identificar os receptores e as arestas empregadas (e seus canais) na tabela de atributos de mensagens.