Ambientes de Suporte à Colaboração em Redes Móveis*

Tatiana Almeida Souza Coelho

Markus Endler

Departamento de Informática - PUC/Rio

{tati,endler}@inf.puc-rio.br

PUC-RioInf.MCC26/02 Outubro, 2002

RESUMO

Esta monografia discute aspectos relacionados a sistemas de suporte à colaboração e, em especial, algumas questões relativas a sistemas de gerência de workflow para ambientes móveis. O conceito de percepção, a questão da política de controle de concorrência, e os problemas decorrentes da mobilidade e das desconexões freqüentes são discutidos. Além de uma comparação entre vários sistemas de gerência de workflow, tais como Exotica, RainMan, CoAct e WorkToDo, esta monografia contém uma proposta de arquitetura de sistema de gerência de workflow para o ambiente de redes móveis sem fio. Nesta proposta, usuários estão espalhados fisicamente e apenas se conectam ao servidor de workflow quando precisam enviar ou receber dados. De acordo com a arquitetura proposta, além de um coordenador central, podem existir ainda diversos coordenadores secundários, que podem delegar tarefas a usuários sob o seu controle. Nesta arquitetura, o conceito de bloqueio de tarefas não é empregado e tarefas podem ser replicadas para a execução por diversos usuários, no intuito de diminuir o tempo de execução do workflow como um todo. As principais contribuições desta arquitetura proposta são a possibilidade de replicação de tarefas e a política otimista de controle de concorrência, que evita que usuários do workflow bloqueiem uns aos outros devido a execuções pendentes de tarefas das quais outras são dependentes.

Palavras-Chave: CSCW, workflow, distribuição, mobilidade, desconexão

ABSTRACT

This work discusses some issues related to collaborative systems, and specifically, some questions related to Workflow Management Systems for mobile environments. The concept of awareness, the concurrency control method, and the problems deriving from mobility and disconnections are discussed. Besides a comparison among several Workflow Management Systems, such as Exotica, RainMan, CoAct and WorkToDo, this report contains a proposal of an architecture for a Workflow Management System for wireless mobile networks. In this proposal users are physically distributed and only establish connection with the workflow server when they need to send or to receive data. According to the proposed architecture, in addition to one central coordinator, there may be several secondary coordinators, which are able to delegate tasks to other users under their control. In this architecture, the concept of locking of tasks is avoided and tasks can be replicated for the execution by many users, aiming to decrease the workflow execution time. The main contributions of the proposed architecture are the ability to replicate workflow tasks and the optimistic concurrency control policy, which provide means for avoiding that workflow users block each other due to pending executions of casually dependent tasks.

Keywords: CSCW, workflow, distribution, mobility, disconnection

*Esta monografia deu origem ao artigo "Gerenciamento de Workflow em Redes com Mobilidade e Conectividade Intermitente"[6], aceito no WCSF 2002 (Workshop de Comunicação Sem Fio), a ser realizado em outubro deste ano, em São Paulo.

Sumário

Lista de Figuras

1 Introdução

O interesse crescente da comunidade científica no estudo de sistemas de CSCW (*Computer Supported Cooperative Work*) ou *groupware* reflete a demanda atual por melhores ferramentas que auxiliam a coordenação e o controle de atividades em grupo.

Diversos sistemas existem atualmente, desde os mais simples, como os sistemas de troca de mensagens eletrônicas (*e-mails*), até os mais sofisticados, que são os sistemas de workflows, bastante utilizados em empresas de comércio eletrônico, de seguro-saúde e no controle de emergências, por exemplo. Também podem ser citados como sistemas de suporte à cooperação os editores multi-usuários de documentos, sistemas de vídeo-conferência, chats e sistemas de discussão.

Sistemas de CSCW podem ser definidos como aqueles que apóiam um número de usuários cooperando para atingir um objetivo particular, como por exemplo a execução de uma tarefa [3, 18]. A colaboração, característica que rege este tipo de sistema, pode acontecer através da comunicação entre os usuários ou por meio do compartilhamento de recursos. Diversas outras definições para sistemas cooperativos podem ser encontradas em [5].

Em [7], Alan Dix define um framework para um sistema de CSCW, como mostra a Figura 1. Os círculos denotados por P representam os participantes do sistema de cooperação. O círculo denotado por A representa os artefatos, ou seja, os recursos compartilhados pelos participantes. A interação, conforme apresenta a figura, pode ocorrer entre participantes/membros do grupo ou mesmo entre participantes e artefatos. A seta tracejada indica que a comunicação entre dois participantes pode ocorrer através de um recurso disponível na rede.

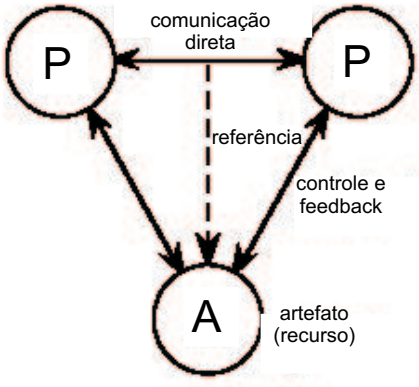

Figura 1: Framework para sistemas de CSCW [7].

A maioria dos sistemas de suporte à cooperação são de natureza distribuída e, por isso, altamente dependentes das plataformas de sistemas distribuídos [18]. Estes permitem que computadores ou dispositivos autônomos compartilhem recursos (software, hardware, dados de aplicação) de forma concorrente e muitas vezes transparente [18]. A transparência neste caso significa que o usuário acessando um recurso não tem conhecimento se este recurso é local ou remoto. No caso de sistemas de cooperação, isso também deve ocorrer. No entanto, ao contrário do que ocorre em sistemas distribuídos tradicionais, na maioria dos sistemas de cooperação usuários devem sempre tomar conhecimento do que os demais usuários estão realizando ou mesmo que de estão cooperando em determinado instante.

Como muitas vezes a colaboração deve ocorrer nas aplicações de CSCW entre pessoas ou grupos de pessoas que não estão em seus locais de trabalho, mas sim em campo, surgiu a necessidade de migrar do ambiente de rede fixa para o ambiente de redes móveis muitas das aplicações de CSCW. Considere, por exemplo, a colaboração entre funcionários da Petrobrás quando da ocorrência de uma emergência, por exemplo o vazamento de óleo em uma das refinarias. A situação ideal é que o funcionário da refinaria possa, através de seu dispositivo móvel e no local do acidente, fornecer as informações corretas da emergência para que as devidas providências sejam tomadas e o plano de ação de emergência seja executado.

Considere, agora, a situação em que vários funcionários estejam cooperando no controle de um acidente. Neste tipo de cooperação é grande a probabilidade de que vários deles estejam acessando e atualizando os mesmos ítens de dados simultaneamente. Se ocorrer uma desconexão inesperada de alguns destes funcionários, os dados compartilhados não poderão mais ser atualizados de modo que todos tenham uma visão consistente dos mesmos.

Dessa forma, o fato de se executar aplicações de CSCW em ambientes de redes móveis traz muitos novos desafios. Contradizendo o objetivo de acesso transparente dos ambientes distribuídos, a natureza dos ambientes de redes sem fio faz com que, inevitavelmente, usuários tomem conhecimento de atrasos ocorridos e de outros problemas inerentes a esse ambiente de execução [7].

Nesse sentido, o objetivo deste trabalho é propor uma arquitetura de software para o suporte à colaboração em sistemas de workflow levando em consideração fatores inerentes a este tipo de sistema e de redes com componentes móveis.

Organização do Trabalho

Este trabalho encontra-se estruturado da seguinte forma. Na seção 2 são levantadas as características e limitações das redes móveis. Na seção 3 são apresentadas as principais abordagens para sistemas de CSCW tradicionais, sua classificação, formas de colaboração e seus problemas quando em ambientes móveis. Na seção 4 são abordadas questões pertinentes a sistemas de workflow. Na seção 5 são apresentados alguns sistemas de CSCW existentes atualmente, com suas características e seus problemas. Na seção 6 é apresentada em detalhes a abordagem proposta. Uma análise comparativa com outros sistemas de suporte de workflow é feita na subseção 6.6. Por fim, a seção 7 apresenta as conclusões deste trabalho.

2 Características e Limitações de Redes Móveis

Redes móveis têm se tornado cada vez mais utilizadas, apesar de todas as suas limitações. O maior apelo deste tipo de rede é o de conciliar a mobilidade do usuário com a possibilidade de acessar dados de qualquer lugar, independentemente de sua localização física. No entanto, a mobilidade traz alguns problemas que não ocorrem em redes fixas. Além da mobilidade, o meio sem fio e a portabilidade dos dispositivos móveis introduzem alguns desafios [14].

Enquanto usuários de dispositivos móveis estão se movimentando, seu ponto de conexão com a rede fixa se altera, o que traz uma série de problemas. O primeiro deles diz respeito à configuração do sistema, que não é estática. Além disso, a carga de trabalho em uma rede, o número de usuários, a qualidade da comunicação e dos serviços por ela prestados, a quantidade de recursos e largura de banda disponíveis para cada usuário variam de forma muito mais rápida e freqüente.

Como usuários estão se movendo e conectados à rede através de diferentes pontos de acesso em momentos distintos, existe um componente de custo de comunicação que está associado à necessidade de se encontrar cada usuário. Ainda, como consultas muitas vezes são dependentes da localização, o tempo de resposta varia conforme a situação da rede que está servindo o usuário no momento de sua execução.

De fato, a mobilidade também implica em questões de segurança e autenticação, porque usuários estão se movendo entre redes distintas. Um usuário pode, por exemplo, tentar se passar por um outro usuário em uma rede na qual este último já esteve conectado.

No que tange questões relacionadas ao meio sem fio, dispositivos móveis contam com fraca conectividade, que muitas vezes é interrompida abruptamente ou mesmo voluntariamente pelo próprio usuário. Desconexões voluntárias normalmente têm como objetivo reduzir o gasto de energia ou o custo de manter a comunicação sem fio. No caso de celulares, o custo é calculado em função do tempo de conexão, enquanto em outros casos o custo é proporcional à quantidade de informação enviada durante a conexão.

Os momentos de desconexão muitas vezes podem ser previstos, por exemplo quando o usuário percebe que o dispositivo está com pouca carga de bateria. No entanto, o número de desconexões involuntárias é muito maior. Dessa forma, é essencial a uma rede móvel prover ao usuário mecanismos de suporte a operações desconectadas, permitindo que o usuário possa acessar os dados de que necessita mesmo desconectado da rede.

Outro fator de importância em uma rede móvel é o tempo total gasto na comunicação: tempo gasto no tráfego de informação pela rede fixa mais o tempo gasto no tráfego na rede sem fio. Este tempo depende, dentre outros fatores, da latência da rede, da largura de banda disponível (principalmente quando a quantidade de informação a ser transmitida é bastante alta), do volume de *buffering* e processamento nos diversos pontos da rede, da própria velocidade do usuário em realizar uma ação e do fato de haver ou não interferências na rede no momento da transmissão. Ainda, o tempo de resposta pode ser bastante variável, dada a configuração dinâmica da rede sem fio.

Quanto aos próprios dispositivos, eles são bem mais frágeis, mais suscetíveis a estragos e com menor capacidade de processamento e armazenamento do que qualquer outro dispositivo fixo de comunicação. Isso pelo fato de necessitarem ser pequenos e leves o suficiente para serem realmente portáteis. Além disso, os recursos disponíveis são limitados, por exemplo a quantidade de memória, o tamanho do espaço de tela e a energia

disponível.

A questão da adaptação em redes sem fio também se apresenta como um desafio. A dinamicidade do status de uma topologia de rede faz com que o usuário do dispositivo móvel necessite se adaptar às condições do ambiente. Se ele possuir em seu aparelho um mecanismo de cache para suprir necessidades provenientes dos momentos de desconexão, políticas de gerenciamento de cache devem ser adotadas.

Porém, o caching de dados em dispositivos móveis não é simples porque a quantidade de espaço de armazenamento disponível nos dispositivos é bastante limitada. Além disso, como a sua capacidade de processamento e o tempo de resposta são bem menores do que em dispositivos de uma rede fixa, algoritmos de caching devem ser eficientes.

Muitas vezes, problemas de tempo e comunicação em redes móveis podem ser resolvidos alterando-se a natureza das tarefas a serem executadas ou simplesmente adequando-as ao tempo de processamento disponível. Por exemplo, se o tempo gasto para a execução remota de uma tarefa for inviável para o processo como um todo, uma alternativa é processá-la localmente no dispositivo do usuário.

Essas são algumas das principais características das redes sem fio e suas mais importantes limitações. Mas e se considerarmos que o usuário não está trabalhando individualmente, mas sim em cooperação com outros usuários? Neste caso os problemas são ainda maiores.

3 Abordagens para CSCW Tradicional

Nesta seção são discutidas questões relativas às abordagens tradicionais para CSCW. É apresentada a classificação dos sistemas de CSCW encontrada na literatura e uma discussão mais detalhada de algumas aplicações mais comuns. Em seguida são apresentados os problemas encontrados quando os sistemas de CSCW são usados em ambientes de redes sem fio.

3.1 Classificação de Sistemas de CSCW

Tom Rodden em [17, 18] apresenta uma classificação dos sistemas de CSCW que se baseia em dois fatores referentes aos sistemas cooperativos: a forma de cooperação/interação e a distribuição geográfica dos usuários.

Segundo o critério de classificação que leva em consideração a forma de interação entre os usuários, sistemas de CSCW podem ser:

- totalmente síncronos: são sistemas do tipo vídeo-conferência em tempo real que necessitam da participação simultânea de todos os usuários. Neste tipo de sistema as atualizações ou quaisquer outros tipos de ações dos usuários devem ser instantaneamente visíveis pelos demais usuários;

- totalmente assíncronos: são sistemas de mensagens cooperativos, como sistemas de e-mail, e sistemas de conferência, nos quais usuários compartilham documentos de forma independente uns dos outros;

- sistemas mistos: sistemas que suportam tanto a cooperação síncrona quanto a cooperação assíncrona. Por exemplo, pertencem a essa categoria os sistemas de co-autoria de documentos, nos quais documentos podem ser editados por usuários em momentos distintos ou simultaneamente.

A Figura 2 apresenta algumas classes de sistemas de CSCW que se enquadram em cada um dos tipos de cooperação descritos.

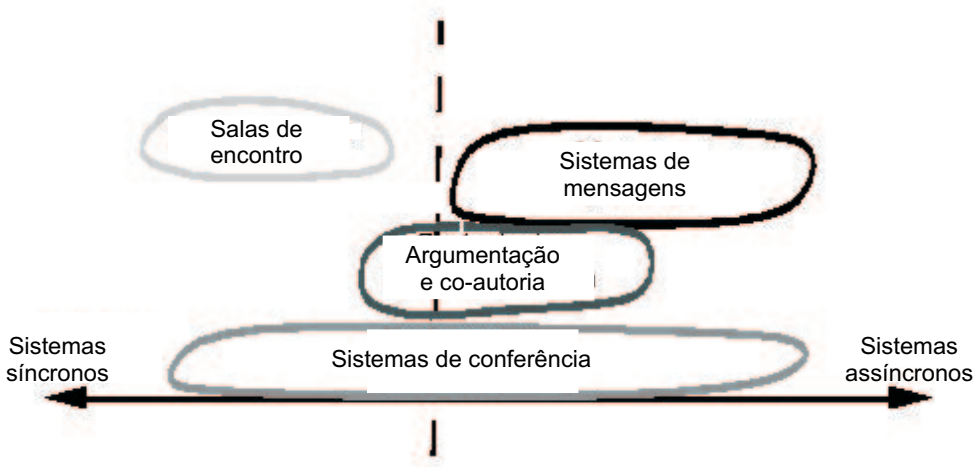

Figura 2: Formas de cooperação em sistemas de CSCW [18].

De acordo com o segundo critério de classificação de Rodden, a distribuição geográfica dos usuários, sistemas cooperativos podem ser considerados:

- remotos: neste tipo de sistema, usuários estão colaborando entre si através de acessos remotos;
- co-localizados: neste caso, a colaboração se dá através do compartilhamento de elementos em um local de encontro comum.

Na Figura 3, Rodden apresenta o eixo de distribuição geográfica, colocando em cada um dos extremos os dois tipos de sistemas considerados e em toda a sua extensão exemplos de sistemas. Ao longo deste eixo, duas divisões a mais foram identificadas: sistemas virtualmente co-localizados e sistemas localmente remotos. Sistemas virtualmente co-localizados são similares aos sistemas co-localizados, mas os usuários não precisam estar em um mesmo local para cooperar. Sistema de vídeo-conferência é um exemplo de sistema virtualmente co-localizado. Sistemas localmente remotos, por sua vez, são aqueles que contam com grande largura de banda para que usuários tenham acesso em tempo-real aos dados através de técnicas de tela compartilhada.

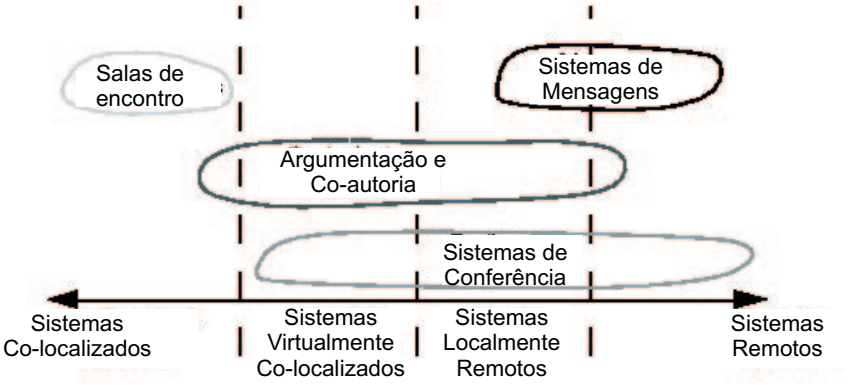

Figura 3: Distribuição geográfica em sistemas de CSCW [18].

Segundo descrito em [7], para se caracterizar de forma mais precisa os sistemas de CSCW em termos de cooperação (eixo de tempo) e da distribuição geográfica (eixo de espaço), é preciso levar em consideração a escala de tempo sobre a qual os usuários sincronizam suas atividades. Em sistemas síncronos, esse tempo é da ordem de segundos. No entanto, em sistemas assíncronos este tempo é da ordem de horas ou até mesmo dias.

3.2 Classes de Sistemas de CSCW

Nesta subseção são abordadas em mais detalhe cada uma das classes de sistemas de CSCW [17] que apareceram nas Figuras 2 e 3.

Sistemas de Mensagens

Sistemas de mensagens são os sistemas de CSCW mais antigos. A sua grande utilização fez com que diversos padrões de estruturação de mensagens fossem criados, como por exemplo o padrão EDI (*Eletronic Data Interchange*) e, mais recentemente, o XML/EDI. São sistemas assíncronos, nos quais os usuários cooperam remotamente, através da simples troca de mensagens eletrônicas.

Sistemas de Conferência

Em sistemas de conferência, usuários interagem através de um espaço de informação compartilhado, como mostra a Figura 4. Usuários podem tanto se comunicar por meio deste espaço, quanto diretamente pela troca de mensagens. A interação pode ser síncrona, em tempo-real, ou assíncrona, em períodos de tempo extensos.

Sistemas de conferência tradicionais são tipicamente formados por um número de grupos (as chamadas conferências), cada um dos quais possui um conjunto de usuários que se comunicam através da troca de mensagens textuais ou multimídia. Neste tipo de sistema, usuários podem se inscrever apenas em seus grupos de interesse, já que cada tópico é abordado por um grupo individualmente. Pode ser citado como exemplo de sistema deste tipo um sistema de suporte a educação, como o AulaNet, em que cada aluno faz matrícula apenas nas disciplinas de interesse.

Figura 4: Interação em sistemas de conferência [17].

Atualmente, sistemas de conferência podem ser classificados de acordo com os dados que são compartilhados no espaço de informação. Os primeiros sistemas de conferência compartilhavam apenas dados textuais, mas com a evolução da tecnologia de tempo-real e da tecnologia multimídia, sistemas deste tipo passaram a fazer uso de uma diversidade de mídias e, por isso, são denominados sistemas multimídia de conferência.

Os sistemas multimídia de tempo-real se utilizando de telas compartilhadas e de dados de áudio e vídeo, tornaram possível que usuários trabalhando local ou remotamente cooperassem de modo síncrono.

Salas de Encontro

Sistemas de "sala de encontro"(os chamados *meeting rooms*) antigamente consistiam de uma sala de conferência (para discussão, debate, encontro) com uma grande tela onde era projetado o vídeo da conferência, um ou mais computadores, terminais de vídeo e um terminal de controle, que organizava o processo de colaboração (Figura 5). Neste caso, os usuários em cooperação se encontravam reunidos em um mesmo local físico.

Atualmente estas salas de encontro são praticamente todas virtuais, de modo que os usuários em colaboração se encontram espalhados fisicamente pela rede. Dois ou mais usuário podem "se encontrar" nestas salas e trocar mensagens eletrônicas. É o que chamamos de *chats*. Muitos deles são classificados por tópicos de interesse ou pela idade dos participantes.

Sistemas de Co-autoria e de Discussão

A classe de sistemas de co-autoria e de discussão representa uma classe de sistemas que visa dar suporte e representar a negociação e discussão em grupos de trabalho. Por exemplo, em sistemas de co-autoria de documentos o suporte à colaboração permite que o resultado final do trabalho seja um documento que espelhe as modificações efetuadas por cada um dos membros do grupo. Neste caso a cooperação pode ser assíncrona e a

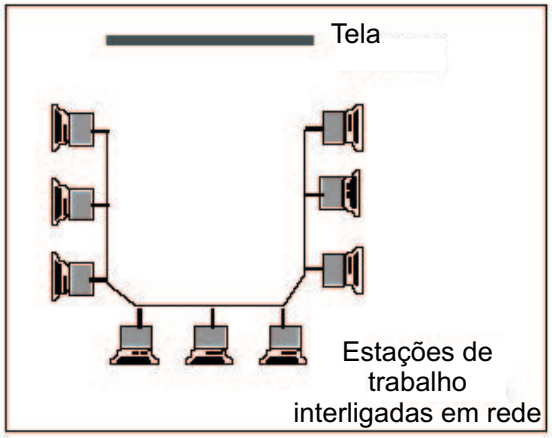

Figura 5: Topologia de uma "sala de encontro": usuários encontram-se em um mesmo espaço físico [17].

localização dos membros em colaboração se torna irrelevante.

Em sistemas de discussão, os participantes também estão espalhados pela rede e podem trocar idéias assincronamente. Neste caso, o sistema é como um espaço de discussão, no qual usuários colocam suas idéias a respeito de um determinado tema.

Dessa forma, as classes de sistemas de CSCW podem ser representadas em um gráfico bi-dimensional com eixos Localização e Interação, de modo que cada classe pode ser representada independentemente em função de cada critério, uma vez que estes eixos são ortogonais. A Figura 6 apresenta o espaço de classificação para sistemas de CSCW mostrado em [17]. Exemplos de sistemas em cada uma das classes podem ser encontrados em [17].

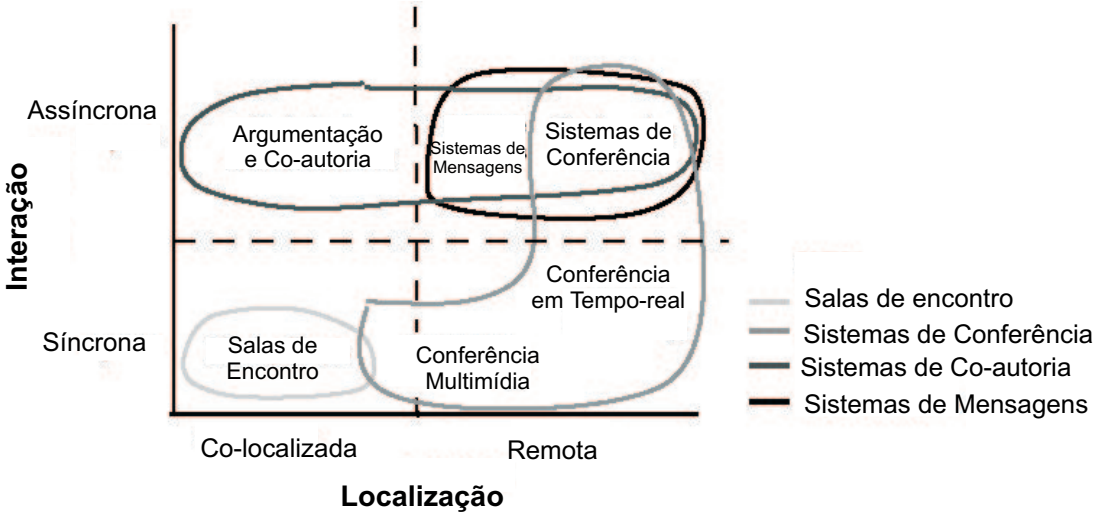

Figura 6: Espaço de classificação de sistemas de CSCW [17].

Existe ainda uma outra classe não citada em [17], mas que tem sido atualmente bastante estudada e que envolve várias áreas de conhecimento. Esta classe, a classe de work-

flows, denota sistemas que visam automatizar o fluxo de tarefas e dados de um processo, a fim de melhorar a eficiência das organizações. Um workflow pode ser definido como um modelo computadorizado do processo de negócio, o qual especifica os passos de cada atividade, a ordem e a condição nas quais cada uma deve ser executada, o fluxo de dados entre estas atividades e as ferramentas/recursos utilizados por cada uma delas. Sistemas de workflow são sistemas mistos, onde a cooperação pode ser síncrona ou assíncrona, e que compartilham recursos, por exemplos documentos de texto. Com relação à natureza geográfica, sistemas de workflow são sistemas remotos, uma vez que membros do grupo em colaboração encontram-se espalhados remotamente pela rede à qual estão conectados.

3.3 CSCW e Sistemas Distribuídos

A maioria dos sistemas de CSCW possuem natureza distribuída e, portanto, dependem da tecnologia de distribuição oferecida. A análise aqui apresentada leva em consideração os dois fatores de classificação de sistemas de CSCW já citados: forma de colaboração e distribuição geográfica.

Sistemas distribuídos têm sido considerados em termos de suporte à colaboração entre vários computadores conectados via rede. Cooperação, neste sentido, refere-se ao quanto estão relacionados os computadores no sistema distribuído. Esta cooperação varia de forma significativa de sistema para sistema.

Na maioria dos sistemas cada estação tem a sua autonomia. Uma vez que o aumento da autonomia de cada estação em um sistema distribuído implica em uma maior dificuldade de sincronização, o grande desafio é definir como cooperação e autonomia podem se relacionar. Nesse sentido, foram definidas as seguintes classes de sistemas, de acordo com o grau de acoplamento entre os serviços disponíveis:

- sistemas autônomos com capacidade de troca de mensagens: neste caso, cada estação do sistema distribuído é autônoma e coopera assincronamente com as demais através da troca de mensagens, geralmente textuais;

- sistemas de compartilhamento de recursos: neste tipo de sistema, cada estação tem acesso aos recursos compartilhados do sistema (por exemplo, pool de processos, impressora), mesmo quando eles são remotos, mas cada estação possui completa autonomia;

- sistemas operacionais distribuídos: são os sistemas operacionais que gerenciam os dispositivos de um ambiente distribuído, de modo que cada estação tem uma visão idêntica dos recursos distribuídos (transparência de distribuição). Eles representam o suporte máximo disponível para cooperação.

Sistemas CSCW síncronos requerem alto nível de cooperação, enquanto que sistemas assíncronos tendem a ser mais autônomos, podendo realizar a cooperação (troca, envio, recebimento de informação) em momentos seguintes à realização das ações. Sistemas tradicionais de cooperação assíncrona normalmente necessitam apenas das funcionalidades oferecidas pelos sistemas de troca de mensagens. A Figura 7 apresenta uma comparação entre os modelos de cooperação, de acordo com Rodden e Blair.

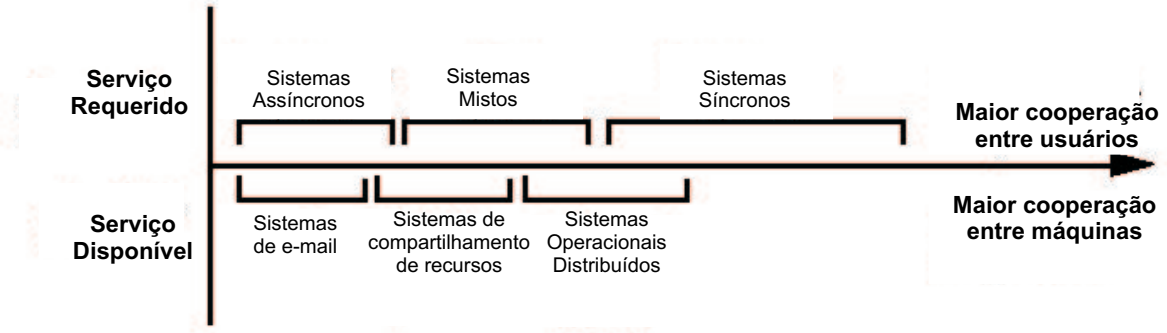

Figura 7: Comparação entre modelos de cooperação [18].

Observe na Figura 7 que quanto maior a cooperação entre usuários, mais difícil é a cooperação entre as máquinas da rede, ou seja, os sistemas distribuídos tornam-se cada vez mais complexos a fim de suprir as necessidades impostas pelos usuários em cooperação. Por exemplo, pela figura fica claro que sistemas de compartilhamento de recursos são mais complexos do que sistemas de e-mail, justamente porque neles existe a necessidade de alto nível de cooperação entre as máquinas.

Limitações ainda existem no que se refere ao suporte de rede aos sistemas de CSCW nos quais a cooperação ocorre sincronamente, principalmente pelo fator largura de banda. Na subseção 3.4 são apresentados alguns problemas dos sistemas de CSCW quando instalados em redes móveis, as quais possuem uma série de restrições se comparadas com as redes fixas.

Controle e Transparência em Sistemas de CSCW

Além dos dois fatores de classificação de sistemas de CSCW já apresentados na subseção 3.1, um outro fator pode ser utilizado para classificá-los: o grau de controle do sistema, que indica o nível de autonomia que o sistema oferece aos usuários.

Em sistemas operacionais distribuídos, a questão da distribuição é normalmente transparente para o usuário (transparência de distribuição). Deste modo, usuários acessam recursos e trabalham sem se preocupar com a questão da distribuição e com possíveis problemas decorrentes dela. Um arquivo pode, por exemplo, estar sendo acessado por um usuário, mas este arquivo não estar local.

Apesar de um sistema de CSCW precisar deste tipo de transparência, nele cada usuário deve ter a percepção do que está ocorrendo, uma vez que o trabalho é cooperativo. Se usuários não têm a percepção do sistema, então eles não podem cooperar entre si e ter controle sobre a forma de cooperação. A Figura 8 apresenta uma tabela com as várias formas de transparência em sistemas distribuídos, as quais visam esconder do usuário os problemas inerentes ao ambiente distribuído.

Se todas essas formas de transparência forem empregadas no sistema distribuído, ele não poderá atender os requisitos de controle dos sistemas de CSCW. Abordagens alternativas foram criadas no sentido de permitir que sistemas distribuídos não sejam por completo transparentes. São elas:

- abordagem de não-transparência: neste caso, apesar de se oferecer maior flexibi-

Transparência	Questão Central	Resultado da Transparência
Localização	A localização de um objeto em um ambiente distribuído	Usuário não tem conhecimento da localização dos serviços
Acesso	O método de acesso aos objetos em um sistema distribuído	Todos os objetos são acessados do mesmo modo
Migração	A relocação de um objeto em um ambiente distribuído	Objetos podem ser movidos sem que o usuário tome conhecimento
Concorrência	Acesso compartilhado aos objetos de um ambiente distribuído	Usuários não têm que lidar com problemas de acesso concorrente
Replicação	Manutenção de cópias de um objeto em um ambiente distribuído	Sistema lida com consistência de cópias de dados
Falha	Falha parcial em um ambiente distribuído	Problemas de falha são escondidos do usuário

Figura 8: Formas de transparência em sistemas distribuídos [18].

lidade no controle, desenvolvedores de aplicações têm sua carga de trabalho aumentada por necessitarem gerenciar toda a distribuição, incluindo mecanismos de recuperação de falhas;

- abordagem de transparência seletiva: neste caso, o usuário pode escolher as formas de transparência que deseja utilizar.

No entanto, nenhuma das duas abordagens citadas resolve o problema de controle em sistemas de CSCW. Este tipo de sistema requer, dessa forma, uma abordagem especial para a qual *awareness* é um fator importante.

Dessa forma, para que usuários possam cooperar entre si, eles devem estar cientes de como estão se comportando os demais usuários e os dados no sistema distribuído. A isso damos o nome *awareness*, ou percepção do ambiente.

A transparência de um sistema distribuído quando no contexto de sistemas de CSCW deve ser vista sob dois pontos: primeiro, ela é necessária com relação ao acesso aos recursos, de forma que o usuário não precisa saber onde está o recurso que está utilizando; por outro lado, o sistema deve tornar o usuário capaz de perceber as atividades dos demais usuários no sistema, de modo que ele possa reagir às mudanças por eles realizadas.

Em [7], Alan Dix distingue três formas de percepção:

- percepção de quem está no sistema: cada membro em um sistema cooperativo deve ter conhecimento dos demais membros em cooperação;

- percepção do que foi feito por cada membro: as atualizações realizadas sobre os dados ou sobre o próprio sistema por um dos membros deve sempre ser visível aos demais, seja pela comunicação direta, seja pela comunicação por meio dos artefatos (Figura 1). Mesmo que essa percepção não ocorra no momento exato da atualização, ela deve ocorrer em algum momento posterior;

- percepção de como a atualização foi feita: deve ser possível a cada membro compreender como ocorreu a atualização de um objeto por outro membro.

Problemas maiores surgem quando esses sistemas de CSCW são levados para o ambiente de redes móveis, onde desconexões são freqüentes e o canal de comunicação instável em qualidade. Neste caso, uma quarta forma de percepção torna-se necessária: a percepção do canal de comunicação.

3.4 Problemas dos Sistemas de CSCW em Redes Móveis

O grande desafio dos sistemas de CSCW para redes móveis é a manutenção da consistência dos dados (artefatos/recursos compartilhados) e a ressincronização de estados após os períodos de desconexão.

Por exemplo, considere um grupo de trabalho operando em modo conectado. Ocorre um problema repentino na rede e um (ou mais de um) dispositivo perde a conexão. A partir deste momento, o usuário neste dispositivo passa a trabalhar no modo desconectado, sendo suas atualizações feitas somente no cache local. No momento da reconexão, os dados deste cache devem ser sincronizados com os dados dos demais usuários que permaneceram conectados à rede. Se conflitos forem detectados, todos os usuários envolvidos devem tomar conhecimento, para que não ocorram conflitos em cascata.

Em [19], por exemplo, é citado o exemplo de uma aplicação de agenda compartilhada. Considere que o sistema de agenda compartilhada é composto de um banco de dados representando os locais onde encontros/reuniões podem ser agendados e um banco de dados privado para cada membro, contendo os horários já agendados por ele.

Nesse contexto, considere que dois grupos distintos desconectados marcam um encontro na agenda na mesma sala e no mesmo horário, já que um grupo não tem conhecimento do outro grupo. No entanto, devido à coincidência de local e horário, um dos dois compromissos posteriormente deverá ser desmarcado. O sistema deve ser capaz de informar todos os membros afetados, para que estes atualizem seus próprios bancos de dados. Além disso, conflitos em cascata poderão surgir caso estes membros, quando avisados do conflito, já tenham marcado novos encontros. O grande problema está no fato de que, como o trabalho é cooperativo, qualquer atualização de um dado por um membro do grupo de cooperação pode afetar o trabalho de outro membro do mesmo grupo.

Dessa forma, os problemas relativos ao ambiente móvel afetam qualquer tipo de sistema/aplicação neste ambiente. Pelo fator mobilidade trazer consigo a possibilidade de ocorrência de desconexões, mecanismos de cache devem ser implantados em cada um dos dispositivos móveis. Se existe caching, é fundamental existirem mecanismos de controle de atualização e de controle de acesso. Se estes mecanismos forem tais que todo usuário tem completo poder de leitura e atualização sobre os dados em seu cache local, mecanismos eficientes de detecção e resolução de conflitos devem ser empregados. Além disso, como espaço de disco em cada dispositivo é limitado, técnicas de gerenciamento de cache específicas devem ser implementadas.

No entanto, além de todos esses mecanismos, existem ainda outros fatores que devem ser levados em consideração quando da construção de sistemas de CSCW para ambientes móveis. A questão temporal é um deles. Como cada usuário trabalha para alcançar um objetivo comum a todo o grupo, o tempo de realização de cada tarefa por cada usuário e da percepção do que está acontecendo pelos demais é de fundamental importância. Por exemplo, se uma tarefa T_1 possui um tempo máximo em que deve ser executada e se outra tarefa T_2 a ser executada por outro usuário depende da primeira, problemas podem surgir

se a notificação sobre término de T_1 não puder ser transmitida aos demais usuários em cooperação no tempo descrito.

Desta forma, arquiteturas de sistemas de cooperação (e workflow) em ambientes móveis devem permitir interações assíncronas. Uma abordagem flexível para a reconciliação entre processamento local autônomo (durante períodos de desconexão) e processamento síncrono (nos períodos de conexão) é um ponto-chave no projeto de um sistema de cooperação para ambientes de redes móveis [8].

3.4.1 O Problema de Bloqueio

Em qualquer sistema distribuído é de extrema importância a política de acesso e atualização dos dados compartilhados. Essa política determina o mecanismo de controle de concorrência empregado. Se cada membro de um grupo de trabalho tem seu próprio cache, em momentos de desconexão provavelmente mais de um membro estará atualizando o mesmo item de dados. Caso a política de atualização seja otimista, conflitos devem ser tratados no momento da reconexão. E se este conflito detectado estiver vinculado a um valor de variável (ou estado) que foi modificado com diferentes valores por dois ou mais membros do grupo, provavelmente a resolução do conflito precise ser feita manualmente, com intervenção humana.

Dessa forma, na política otimista empregada sobre sistemas cooperativos em ambientes móveis a probabilidade de que vários membros atualizem o mesmo item de dado durante desconexão é bastante alta. Isso porque, como os ítens de dados não estão bloqueados, durante desconexão cópias do mesmo item podem estar em caches de dispositivos distintos, podendo sofrer alterações distintas.

A outra alternativa é uma abordagem pessimista de atualização, através do uso de bloqueios. De acordo com essa abordagem pessimista, que evita a ocorrência de atualizações simultâneas que possam gerar conflito, cada item de dado a ser atualizado por um membro é previamente bloqueado e somente liberado depois que a atualização tiver sido completada. No entanto, no caso de aplicações em ambientes de redes móveis, essa abordagem pode levar a longos períodos de inatividade por parte dos usuários que esperam pela liberação do item de dado para continuar a execução de suas tarefas.

Considere um membro atualizando um item de dado quando conectado. Esse item deve estar, desta forma, bloqueado para a sua atualização. Antes que esta seja completada, o dispositivo no qual o membro está atuando se desconecta da rede e demora muito a voltar. Enquanto isso, nenhum dos demais membros que permaneceram em conexão poderão ter acesso a este item de dado.

Assim, a abordagem pessimista em redes móveis compromete a disponibilidade dos dados, tão importante nos sistemas de cooperação. Nestes sistemas, os dados devem sempre estar disponíveis aos membros do grupo de trabalho, mesmo quando operam em modo desconectado.

3.4.2 O Problema da Reintegração

A reintegração ocorre quando um ou mais membros da rede antes desconectados voltam a se conectar. Como os períodos de desconexão são muito freqüentes em redes móveis, é

desejável que a detecção e a resolução de conflitos possam ser feitas de forma automática pelo próprio sistema. Para isso, precisa-se:

- definir a granularidade dos dados a serem considerados no momento da reintegração. Por exemplo, a reintegração pode ocorrer sobre uma tabela de uma única vez ou, por exemplo, sobre cada campo de uma tabela separadamente. O nível de granularidade afeta o tempo necessário para efetuar a reintegração;

- manter sempre atualizada a informação de que membro do grupo tem acesso a qual item de dado, para que a resolução de conflitos após a reconexão dos membros em cooperação torne consistentes os valores dos ítens de dados.

Quando o processo automático não é possível, a resolução de conflitos deve ser feita manualmente. Para isso, é preciso comparar os valores/estados dos dados replicados, o que muitas vezes ocorre pela comparação do conteúdo e do *timestamp*, e identificar/produzir um novo valor consistente. Considerando apenas duas réplicas de um item de dado, o problema da resolução de conflitos consiste em [7]:

- identificar qual das réplicas é mais recente;

- verificar se ambas sofreram alteração;

- verificar se as atualizações são independentes, isto é, se é possível criar uma nova versão (valor) contemplando ambas as atualizações;

- verificar se cada item de dado aparentemente não replicado é um novo item ou se foi removida uma réplica de determinado item de dado.

Para que as réplicas de dados voltem a estados consistentes, é preciso definir mecanismos e algoritmos para se comparar valores de dados distintos de um mesmo item e derivar um novo valor consistente[1]. Para isso, muitas vezes torna-se necessário comparar cada um dos valores com o último valor consistente armazenado pelo sistema. Dependendo do tipo de dado, o novo valor consistente pode ser o valor de uma das réplicas ou uma combinação dos valores.

Dessa forma, a resolução de conflitos feita manualmente no momento da reintegração é aceitável se o tempo de desconexão tiver sido muito grande, o que implica em conflitos mais difíceis de serem resolvidos. Para desconexões curtas e freqüentes, é desejável que o sistema consiga efetuar a reintegração de forma automática. O sistema pode, por outro lado, até mesmo tolerar um certo nível de inconsistência entre réplicas mesmo quando os dispositivos estão conectados, no intuito de diminuir o tráfego na rede decorrente de um processo de reintegração. Neste caso, no momento da reintegração as inconsistências devem ser resolvidas.

A seção seguinte apresenta questões relacionadas com a gerência de workflows, que são um tipo específico de sistemas de colaboração.

[1] O critério de consistência de um item de dado é definido pela aplicação

4 Aspectos Relevantes no Projeto de Sistemas de Workflow

O trabalho aqui apresentado está focado no estudo de sistemas de workflow para computação móvel e, por isso, a partir deste ponto somente esta classe de sistemas de CSCW será estudada.

Um sistema de workflow típico é composto dos seguintes elementos: uma máquina de workflow que gerencia todo o processo de execução de uma instância de workflow e a distribuição de tarefas, um banco de dados onde estão armazenadas as informações relativas à cada tarefa e módulos de interação com o usuário.

São basicamente seis os aspectos relevantes no projeto de sistemas de gerência de workflows, apontados em [12]. São eles:

1. recuperação de falhas em sistemas de gerência de workflow distribuídos;
2. navegação em workflows e atividades compensatórias para o caso de ocorrência de falhas;
3. garantia de alta disponibilidade de dados;
4. tratamento de usuários móveis e temporariamente desconectados;
5. coordenação distribuída;
6. modelos de transação avançados.

Com relação à recuperação de falhas, a meta é que um sistema de workflow esteja continuamente disponível, mesmo quando da ocorrência de falhas. Assim, cada dispositivo (e componente do sistema) deve ser capaz de lidar com falhas de comunicação e falhas locais, sem que a execução de qualquer processo de workflow seja interrompida.

Para que uma tarefa do workflow possa ser executada, todos os elementos do sistema devem estar se comunicando. Se a falha ocorrer no banco de dados, o mecanismo de replicação pode resolver este problema. Se os dados necessários para a execução da tarefa estiverem replicados no dispositivo em que ela está sendo executada, a falha no banco não implica na interrupção da execução. Se a falha ocorrer no sistema de gerência de workflow, as operações sob responsabilidade do sistema devem ser executadas em partes que estão em funcionamento. Para que isso seja possível, múltiplas conexões devem ser permitidas entre os componentes do sistema.

No que diz respeito ao segundo aspecto, navegação em workflows e atividades compensatórias em caso de falhas, o problema está no fato de que diversas instâncias de execução de um mesmo workflow podem estar rodando simultaneamente. Neste caso, cada instância pode estar em um ponto distinto da execução com relação às demais instâncias. No entanto, alguns erros não previstos podem ocorrer, exigindo que uma determinada execução seja abortada e, do mesmo modo, todas as tarefas já finalizadas desta instância. Assim, estas tarefas devem ser realizadas novamente quando a execução do workflow for reiniciada. Deste modo, sistemas de workflow devem ter mecanismos que permitam que mesmo diante de erros a execução de uma instância continue ou que a execução

seja recompensada através da realização de outras atividades, as chamadas atividades compensatórias.

O terceiro aspecto diz respeito à principal funcionalidade que um sistema de workflow deve oferecer: disponibilidade ininterrupta aos dados necessários para a execução das tarefas (sejam eles documentos ou variáveis). Na maioria dos casos, esta característica é alcançada por meio do mecanismo de replicação: os dados são replicados nos dispositivos de cada membro do grupo em cooperação. Além de replicação, também são necessários mecanismos que garantam que mesmo diante de reconfiguração dinâmica do sistema, por exemplo de incorporação/exclusão de servidores, o sistema permaneça acessível.

O quarto aspecto é de grande importância, dado o aumento no número de dispositivos móveis utilizados atualmente para a execução de atividades diversas, em especial aquelas atividades de campo, onde aplicações de workflow são mais usadas. Se o servidor de banco de dados e de workflow são centralizados e os usuários executores das tarefas utilizam dispositivos móveis, duas situações podem ser consideradas. Na primeira delas, os usuários permanecem sempre conectados ao servidor. Na segunda, os usuários apenas se conectam quando precisam receber ou enviar dados ao servidor.

Idealmente os usuários deveriam permanecer sempre conectados ao servidor centralizado do workflow enquanto estivessem em operação, mas devido aos freqüentes momentos de desconexão, às limitações de bateria e ao custo de operação isto nem sempre é possível. Dessa forma, apesar de gerar problemas de coordenação, sincronização e controle de concorrência, a melhor forma de operação é aquela em que usuários permanecem desconectados e somente se conectam quando precisam receber ou enviar dados ao servidor de workflow. A Figura 9 apresenta a arquitetura de um sistema de workflow distribuído em que membros do grupo em cooperação podem estar ligados à rede através de dispositivos móveis.

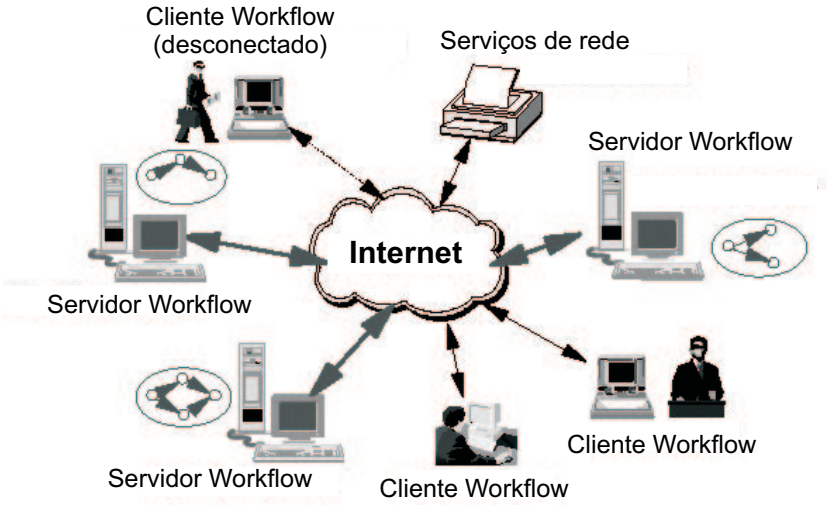

Figura 9: Arquitetura de um sistema de workflow descentralizado.

A coordenação distribuída também é bastante importante: em sistemas de workflow distribuídos, usuários em cooperação encontram-se dispersos geograficamente pela rede. Existem basicamente duas abordagens com relação às formas de percepção: uma delas é empacotar toda a informação pertinente ao workflow (tarefas, recursos associados e

informação de possíveis executores) e seu status de execução, e enviá-la para os diferentes usuários na rede. Essa abordagem pode ser inviável devido ao tamanho do pacote de dados (conjunto de todas as informações a serem enviadas) e também pelo fato de ser necessária a manutenção de diversas cópias deste pacote, para que possa haver execução concorrente. A outra abordagem é pré-compilar o workflow, de modo a se determinar quais usuários em quais dispositivos devem executar quais tarefas, de maneira que a informação a ser transmitida entre eles seja apenas relativa ao status da execução, como por exemplo a informação de todas as tarefas completadas e das tarefas em andamento. Nesta abordagem, se falhas ocorrerem em um dispositivo, suas tarefas podem ser enviadas para outro dispositivo em operação.

O último ponto de pesquisa, os modelos de transação avançados, refletem o fato de que sistemas de workflow são ferramentas baseadas em modelos de transação distribuída. No entanto, para o modelo de workflows são necessários requisitos mais complexos, ainda não tratados pelos modelos de transação.

O *Workflow Management Coalition* (WfMC) [9] propôs uma arquitetura de referência para sistemas de workflow, conforme apresentado na Figura 10.

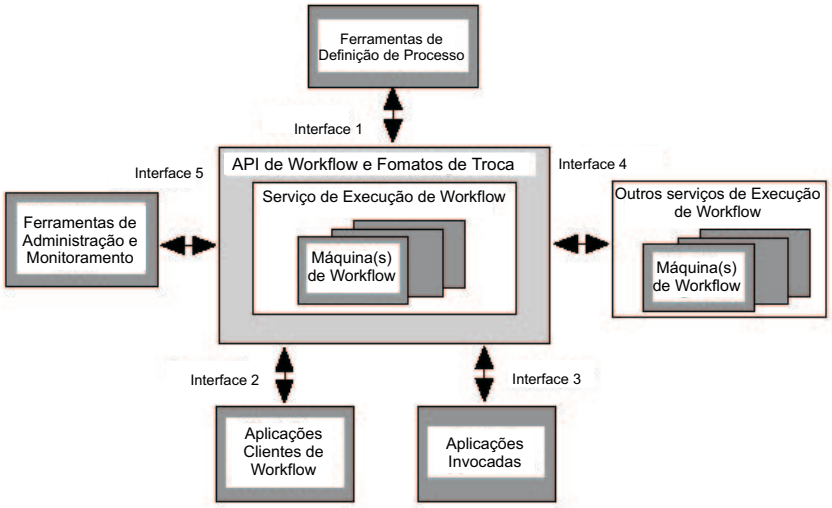

Figura 10: Arquitetura do modelo de referência WfMC [9].

Nessa arquitetura, as ferramentas de definição de processo são usadas para especificar os workflows. As ferramentas de administração e monitoramento são utilizadas para a monitoração do progresso da execução das instâncias dos workflows. As aplicações clientes são aquelas que representam os usuários na execução de suas tarefas do workflow. As aplicações invocadas são aquelas necessárias para a execução de determinadas tarefas. Por fim, o sistema de execução do workflow é o sistema de execução para instâncias de workflows e ativação de tarefas. Este sistema controla cada tarefa no workflow, associando-as a recursos e usuários executores.

Essa arquitetura, porém, reflete um sistema centralizado e pouco flexível para apoiar workflows baseados na Internet, como é o caso de RainMan, e workflows descentralizados. O modelo de referência de WfMC propõe um servidor responsável pela execução dos processos, gerenciamento das informações associadas e distribuição das atividades entre

os membros do grupo em cooperação. Além disso, esse modelo de referência não propõe uma linguagem padrão de especificação de workflows (atualmente cada sistema é definido a partir de sua própria linguagem) e assume que a conexão entre membros do grupo é permanente.

A seção seguinte apresenta alguns projetos e produtos de workflow existentes, procurando focar nos pontos de pesquisa assinalados anteriormente.

5 Sistemas de Workflow

Nesta seção são apresentados alguns sistemas de workflow existentes para ambientes de redes móveis.

5.1 Magi

Magi (*Micro-Apache Generic Interface*) [4] é um framework arquitetural para workflow, baseado no protocolo HTTP e na troca de mensagens, que oferece um conjunto de serviços aos usuários através do carregamento dinâmico de protocolos de serviços.

Algumas características do protocolo HTTP limitam o fluxo de informação entre usuários móveis e desconectados de um sistema de workflow:

- chamadas são sempre iniciadas por um cliente, nunca por um servidor. Deste modo, quando um item de dado é atualizado no servidor, este somente pode propagar a atualização quando o cliente determinar que a atualização é necessária;

- apesar dos servidores sempre terem nomes facilmente identificáveis pelos clientes, o contrário não é verdade, porque clientes não seguem a convenção de nomes dos servidores. E mesmo que seguissem, um servidor não pode iniciar uma conexão com um cliente se este não fez nenhuma requisição. Uma abordagem seria cada servidor armazenar em log as variáveis de ambiente de cada cliente e seu endereço IP. Mesmo assim, pode ocorrer que um servidor não consiga se comunicar com um cliente pelo fato deste estar desconectado ou pelo fato do IP ter sido dinâmico e de outro cliente ter pego este mesmo endereço IP, o que faria com que a mensagem fosse entregue ao cliente errado;

- apesar de atualmente ser possível se conectar à Internet via dispositivos móveis, a conexão ainda é bastante lenta.

No entanto, o uso de HTTP está baseado no fato de que a Web comporta, pelo menos por enquanto, um número variável e bastante grande de usuários e recursos, mantendo a interoperabilidade entre eles. Além disso, como o HTML é independente de conteúdo, Magi pode utilizar nos dispositivos qualquer protocolo.

Uma desvantagem, por outro lado, é a Web não fornecer mecanismos adequados para a colaboração e a coordenação. O que muitos desenvolvedores têm feito é criar uma máquina de workflow baseada em tecnologia de banco de dados, mas com suporte HTTP, de modo que *thin clients* móveis possam acessar dados e tarefas do workflow. No entanto, mudanças são necessárias nos clientes, até mesmo porque a conexão é sempre instável, a

largura de banda baixa e a capacidade de processamento dos dispositivos móveis limitada, dificultando operações em modo desconectado.

A vantagem de se utilizar mecanismos de troca de mensagens para a comunicação assíncrona está no fato destes mecanismos contarem com a característica de *store-and-forward*, ou seja, de permitirem que mensagens que não possam ser enviadas instantaneamente sejam armazenadas para posterior envio. No entanto, quando se trata do envio de mensagens para dispositivos móveis, se o cliente está desconectado uma exceção pode ser gerada no sistema.

Uma outra desvantagem do mecanismo de mensagens está no fato de que uma vez enviada uma mensagem, por exemplo com um arquivo anexado, não é possível saber se o usuário acessou ou não aquele arquivo, e muito menos se ele foi modificado pelo usuário. Além disso, a manipulação do conteúdo a ser disponibilizado na tela em um dispositivo móvel ainda é bastante limitada e a velocidade de transmissão bastante baixa.

Dessa forma, Magi é essencialmente um servidor Apache, que possui o HTTP como seu principal protocolo de comunicação, mas que se utiliza de uma variedade de protocolos para o aumento da diversidade de serviços que pode oferecer e para permitir o acesso de usuários em dispositivos móveis ao sistema de workflow. A Figura 11 apresenta a arquitetura de componentes de Magi.

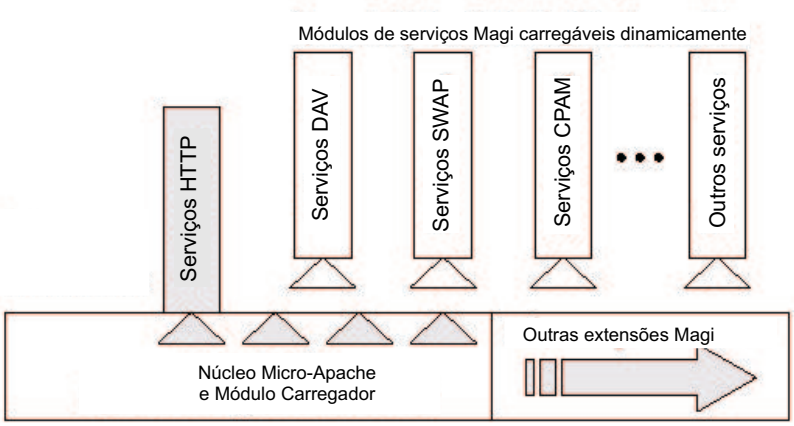

Figura 11: Arquitetura de componentes Magi [4].

Dentre os módulos de serviço dinamicamente carregáveis estão os serviços DAV, SWAP e CPAM. Os serviços DAV são oferecidos através do protocolo WebDAV (*Web-based Distributed Authoring and Versioning*), que compreende um conjunto de extensões do protocolo HTTP, as quais permitem que usuários editem e gerenciem arquivos em servidores Web remotos de forma colaborativa. Maiores informações sobre este protocolo podem ser encontradas em http://www.webdav.org.

Os serviços SWAP são oferecidos pelo protocolo SWAP, o qual reusa vários métodos de extensão de WebDAV para HTTP, definindo extensões adicionais. Os serviços CPAM, por sua vez, são oferecidos pelo protocolo CPAM, que é também uma extensão do protocolo HTTP, com o objetivo de iniciar, finalizar e monitorar a execução de processos em um servidor.

As tarefas a serem realizadas por servidores e clientes são baseadas em eventos que eles

geram. Por exemplo, atividades de workflow podem ser registradas por clientes através de um evento. Do ponto de vista do cliente, um servidor que monitora todo o processo de execução do workflow representa um serviço, como por exemplo um serviço de impressão.

Em Magi, cada dispositivo tem a ele associado um endereço IP estático ou dinâmico. Cada servidor possui a identidade do usuário, uma chave pública única e uma chave privada, e uma lista em XML dos servidores cadastrados e dos serviços que eles suportam. A cada registro de um novo servidor todos os clientes são avisados.

Uma aplicação típica de workflow que pode ser executada através de Magi é a cooperação através do envio sucessivo de um mesmo documento para uma seqüência de clientes. O documento é enviado via mensagem para um cliente e o cliente que o recebeu tem a obrigação de marcar o recebimento e passar para outro cliente o mesmo documento, de modo que no final da troca de mensagens o documento tenha passado por todos os clientes. Dessa forma, o mecanismo de mensagens permite que as tarefas sejam enviadas e executadas por cada cliente de modo assíncrono, mas em colaboração, já que cada cliente envia a mensagem para o próximo. Maiores informações sobre o sistema Magi podem ser obtidas no endereço http://magi.endeavors.org.

5.2 Exotica

Exotica é um projeto do IBM Almadem Research Center, no qual foi desenvolvido o sistema de workflow FlowMark [2, 12, 15].

Nesse sistema, os dados são armazenados em um banco de dados orientado a objeto chamado *ObjectStore*, que não garante que em determinado ponto os dados relativos a uma tarefa em execução não sejam perdidos (esse é o primeiro aspecto a ser considerado com relação ao controle de falhas). As aplicações que estão interagindo com o sistema de workflow podem não estar cientes disso e, portanto, podem deixar de cooperar ou mesmo terminar a execução de uma tarefa, sem levar em consideração o estado do sistema.

Em FlowMark os dados da aplicação não são armazenados juntamente com o sistema de gerência de workflow e, por isso, um gerenciador de dados de aplicação é necessário. Este gerenciador é o Lotus Notes, enquanto que o gerenciador de fluxo de dados de controle é o próprio Exotica/FMQM (*FlowMark on Message Queue Manager*). Este permite que um conjunto de servidores autônomos possam cooperar para completar a execução de uma tarefa. Esta arquitetura foi expandida para incorporar um gerenciador de dados como um conjunto de bancos de dados replicados fracamente sincronizados, retirando da máquina de workflow toda a carga de manipulação destes dados.

Além do controle de perda de dados, a gerência de falhas deve levar em consideração situações em que um dos componentes do sistema de workflow falha. Se o componente que falhou é o banco de dados, mecanismos de replicação podem ser utilizados para minimizar o impacto da falha. No entanto, o maior problema ocorre quando o componente que falhou é parte do sistema de gerência de workflow. Para solucionar problemas provenientes deste tipo de falha, FlowMark emprega a noção de clusters de servidores.

Cada cluster de servidores está vinculado a um banco de dados separado. Como cada banco de dados contém o mesmo esquema, instâncias de workflows/processos podem ser executadas em qualquer cluster. Neste caso, as únicas instâncias afetadas são as que estavam rodando em um cluster que falhou. Instâncias são os workflows/processos em execução. Várias instâncias podem ser executadas a partir de um mesmo workflow.

Com relação a como se comportar diante de erros não previstos na execução de instâncias de workflow, FlowMark utiliza o mecanismo de *forward recovery*, que diz que o sistema continua em execução quando da ocorrência de falhas e somente depois delas se recupera.

No caso de falhas semânticas, que são bem mais difíceis de serem tratadas, FlowMark procura utilizar o mecanismo inverso: *backward recovery*. Utilizando este mecanismo, o usuário determina uma forma de compensação quando da ocorrência de falhas. De mesma importância é a capacidade de se poder mudar o fluxo de controle para uma direção distinta, durante a execução do workflow.

Com relação ao mecanismo de replicação, FlowMark emprega a arquitetura *primary/backup*. De acordo com esta arquitetura, instâncias de um workflow executam em um servidor primário, enquanto todas as ações são também registradas no servidor de backup. Neste caso, se o servidor primário falhar, o servidor de backup passa a assumir o controle da execução das instâncias. Um dos problemas desta arquitetura está no alto custo relativo à sincronização entre as réplicas.

FlowMark define três níveis de prioridade com relação à replicação:

- *hot stand-by*: um processo neste nível de prioridade é completamente replicado, de modo que todas as alterações em seus dados são efetuadas tanto no servidor primário quanto no servidor de backup;

- *cold stand-by*: um processo neste nível é replicado no servidor primário, mas o servidor de backup mantém apenas as informações relativas aos diferentes passos tomados durante a execução daquela instância do workflow (processo). Quando o servidor de backup precisar assumir o papel de *primary*, este deve primeiro rever suas mensagens, a fim de reconstruir um estado de execução atualizado. Processos no nível de prioridade *cold stand-by* têm o reinício de sua execução atrasado pelo servidor de backup, mas, por outro lado, reduzem o atraso decorrente da sincronização das cópias replicadas;

- normal: processos neste nível não são replicados. FlowMark apenas garante que, quando o servidor voltar a funcionar, o processo será executado do ponto em que havia parado.

Com relação à mobilidade, FlowMark permite que operações sejam executadas em modo desconectado. Para que possam operar em modo desconectado, usuários FlowMark devem bloquear a tarefa que desejam executar antes de se desconectar, adotando assim um mecanismo pessimista de controle de concorrência.

Exotica/FMQM: Gerência de Dados de Controle

Exotica/FMQM é uma arquitetura distribuída, que permite que usuários independentes mas em cooperação se comuniquem através de filas de mensagens persistentes [1, 2]. Deste modo, não existe um servidor centralizado onde a informação fica armazenada. Em vez disto, as mensagens trocadas permitem que cada usuário tome a decisão com relação à execução de suas tarefas.

Em FlowMark são os seguintes os componentes principais do modelo de workflow: processos, atividades, conectores de controle, conectores de dados e condições. O processo de

negócio é um grafo acíclico direcionado, no qual os nós representam passos da execução e arestas representam o fluxo de controle e de dados. As atividades são os nós do grafo que representam os passos a serem completados. Os conectores de controle, representados como arestas direcionadas, são utilizados para especificar a ordem de execução das atividades. Os conectores de dados, também representados como arestas direcionadas no grafo do processo, especificam o fluxo de informação de uma atividade para outra na forma de mapeamentos entre os repositórios de dados das atividades (cada atividade tem um repositório de dados de entrada e um repositório de dados de saída). Por fim, condições especificam quando certas eventos podem ocorrer. São três os tipos de condições: condições de transição associadas com conectores de controle determinam se o conector avalia para verdadeiro ou falso; condições de início, as quais especificam quando uma atividade será iniciada; e condições de término, as quais são usadas para especificar quando uma atividade é considerada como terminada.

A Figura 12 apresenta o funcionamento da máquina de workflow de FlowMark.

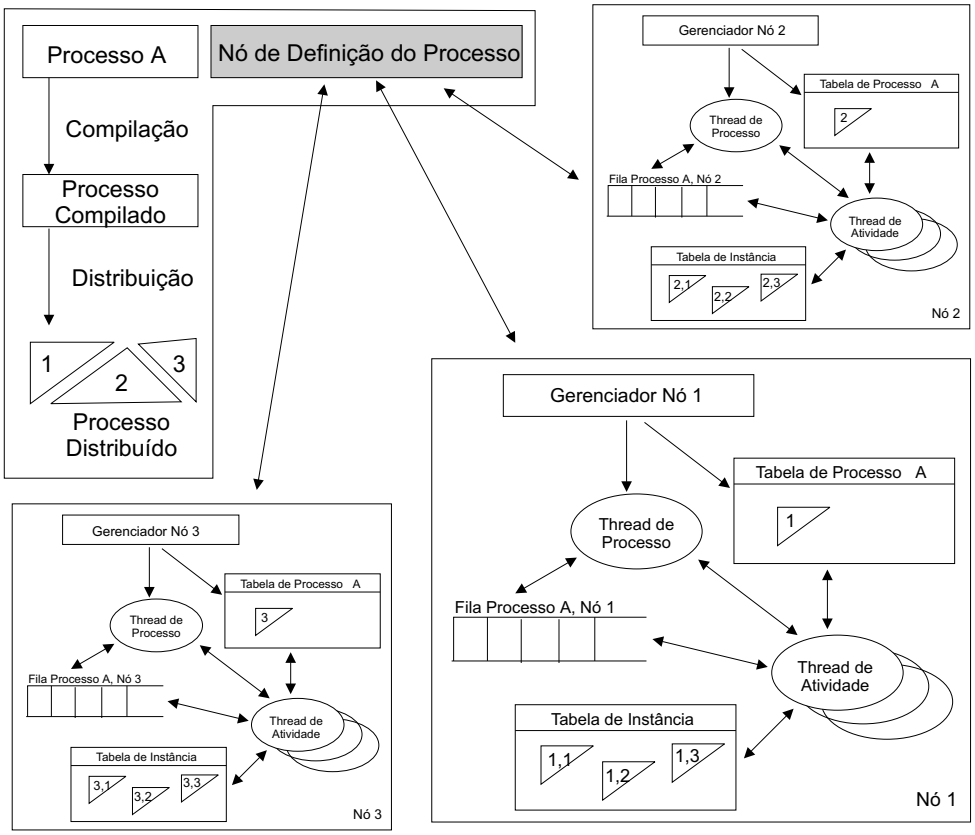

Figura 12: Funcionamento do sistema de workflow FlowMark [1].

Quando um usuário cria um processo a ser executado, ele é compilado para uma representação distribuída e posteriormente dividido em diversas partes, as quais devem ser entregues aos usuários em colaboração (membros do grupo de trabalho), que estão localizados em seus nós de trabalho[2]. Essa divisão do processo é baseada nos usuários

[2]Um nó corresponde a uma máquina remota onde ocorre a execução de partes de uma instância de um workflow.

que serão responsáveis pela execução das atividades.

Assim que recebe sua parte do processo (lista de ítens/atividades que precisa executar), cada nó cria uma tabela de processo para armazenar esta informação (a informação desta parte do processo a ele atribuída) e inicia uma *thread*[3] de processo, responsável pela execução de instâncias deste processo. Apesar de ser mantida em memória por questão de eficiência, cada tabela deve ser armazenada em memória persistente, já que descreve quais ações devem ser tomadas em cada passo da execução do processo. Em geral, esta tabela é uma estrutura *append only*.

A *thread* de processo, por sua vez, cria uma fila de mensagens (inicialmente vazia) para armazenar as mensagens a serem trocadas entre os nós onde estão sendo executadas as partes daquele processo. Durante a execução do processo, a informação relativa às instâncias é armazenada em uma tabela de instância que é manipulada pelas *threads* de atividade. Estas são responsáveis pela execução de atividades individuais dentro de uma instância. Cada *thread* de atividade é criada quando uma instância de atividade torna-se pronta para ser executada. Uma instância de uma atividade representa a unidade de execução da tarefa.

Mensagens são colocadas nas filas através de chamadas PUT e retiradas da fila através de chamadas GET. Estas chamadas são executadas dentro de uma transação, de modo que seu efeito somente é visível e persistente quando a transação é encerrada.

Quando uma instância de um processo é iniciada, mensagens são colocadas, via chamada PUT, nas filas dos nós onde as atividades de inicialização estão localizadas. Dessa forma, quando as mensagens chegam em cada nó, através das filas de mensagens, a *thread* de processo é disparada. Esta *thread* analisa a mensagem, verifica na tabela do processo quais passos devem ser executados, ativando as *threads* de atividades necessárias. Mensagens analisadas não são retiradas da fila, porque são persistentes.

Cada *thread* de atividade, por sua vez, verifica a tabela do processo à procura dos dados relativos a sua tarefa. De posse dos dados, ela cria uma entrada na tabela de instância para esta atividade, na qual ela armazena o progresso da atividade. A condição de início da atividade é então verificada. Se ela não pode ser avaliada, a *thread* de atividade passa a esperar que a *thread* de processo novamente a invoque. Se a condição é falsa, a atividade passa para o estado de terminada. Neste caso, a entrada é retirada da tabela de instância e as mensagens correspondentes à atividade são retiradas da fila do processo. Por outro lado, se a condição de início da atividade é verdadeira, a aplicação correspondente é invocada e à ela são passados os dados necessários à execução. Quando a aplicação termina, os dados de saída são armazenados na tabela de instância.

Neste ponto, a *thread* de atividade avalia a condição de término da atividade. Se ela é falsa, a aplicação é novamente invocada. Caso contrário, a atividade é considerada encerrada e mensagens são colocadas na fila do processo. A entrada é então removida da tabela de instância e todas as mensagens correspondentes àquela atividade são retiradas, via chamada GET, da fila de mensagens do processo.

Gerência de Dados de Aplicação

A idéia principal para a integração do fluxo de controle e o fluxo de dados consiste em

[3]Uma *thread* é uma parte de um programa que pode ser executada independentemente das demais.

estabelecer mecanismos necessários para assegurar que a mensagem colocada na fila de mensagens de um processo com relação à invocação de uma atividade também dispare mecanismos de replicação/distribuição de dados da aplicação (Lotus Notes).

Em Exotica, o nó de controle (máquina da rede responsável pelo controle da execução de processos) representa a parte do sistema correspondente ao Exotica/FMQM, enquanto o nó de dado é a parte do sistema correspondente ao Lotus, o qual fornece acesso aos dados. As atividades manipuladas pelo nó de controle não contém dados de aplicação, mas apenas informação de controle necessária. Essa separação entre nós de controle e de dados traz algumas vantagens para o sistema:

- nós de dados normalmente são compatíveis com as estratégias de gerência de dados de uma companhia, ou seja, não necessitam ser alterados caso as estratégias de gerência o sejam. A introdução de um sistema de gerência de workflow não implica em mudanças das políticas existentes;
- a manipulação de dados complexos pode ser flexível e otimizada;
- juntamente com a noção de clusters de servidores de dados, a separação de nós de controle e de dados proporciona uma arquitetura mais escalável, tolerante a falhas e com períodos mais longos de disponibilidade de dados.

Trabalho Cooperativo Desconectado

Como Exotica utiliza o conceito de bloqueio de tarefas para evitar que mais do que um usuário execute uma mesma atividade, durante períodos de desconexão usuários apenas podem executar atividades bloqueadas. No momento da desconexão todas as tarefas não bloqueadas são removidas da lista de trabalho do usuário. Para a execução da tarefa bloqueada o usuário, representado pela máquina cliente, recupera os dados necessários de seu próprio repositório. O resultado da execução é armazenado localmente até que o usuário volte a se reconectar e tenha acesso ao banco de dados do sistema.

Conflitos gerados por requisições de bloqueio sobre atividades são resolvidos através da serialização que ocorre no servidor de workflow e no banco de dados. Se uma requisição por um bloqueio chega ao servidor após a atividade já ter sido bloqueada por outro usuário, esta requisição é descartada.

Reintegração

Devido ao fato da política de bloqueio de atividades ser empregada em FlowMark, conflitos não ocorrem. No entanto, no momento da reconexão, a máquina de execução do cliente descarta toda a informação que tinha a respeito da lista de trabalho, adquirindo a lista atualizada a partir do servidor.

Contudo, como nem todas as atividades bloqueadas são executadas pelos usuários durante desconexão, a máquina de execução realiza as alterações apropriadas sobre a lista recebida do servidor, incluindo nela as atividades bloqueadas mas ainda não executadas. Estas atividades, independentemente do número de desconexões e reconexões, permanecem bloqueadas até que sejam efetivamente executadas.

Recuperação de Falhas

O Exotica/FMQM permite o suporte à recuperação em caso de falhas, já que mensagens de uma atividade somente são retiradas da fila quando não mais são necessárias. Em caso de falhas que resultam na perda da tabela de instância, um nó de recuperação reinicia a *thread* do processo, a qual encontra as mensagens ainda na fila e, baseada nelas, dispara novamente as *threads* de atividade, as quais reconstróem a tabela de instância. Esta é a razão pela qual as tabelas de instância não necessitam ser armazenadas em memória persistente e pela qual tabelas de processos devem ser armazenadas em memória persistente. Além disso, um log é necessário para manter controle sobre as atividades já completadas e para armazenar mensagens, no caso de um usuário resolver reiniciar uma atividade terminada.

5.3 RainMan

RainMan [13, 15] é um framework de workflow distribuído da IBM projetado em Java para a Internet. Ele está implementado como uma coleção de serviços independentes, e está baseado no framework RainMaker, que define as interfaces para os componentes do sistema.

O framework RainMaker define basicamente quatro elementos no domínio de um workflow:

- *sources*, ou requisitantes de serviços;
- atividades, ou requisições de serviço, as quais são geradas pelos requisitantes;
- executores, que são pessoas, aplicações, organizações ou outras entidades para as quais as atividades são endereçadas. São também chamados de provedores de serviços;
- tarefas, que são a implementação das requisições de uma atividade gerada por um requisitante.

Para que o sistema de workflow funcione de acordo com seus quatro elementos, são definidas três interfaces distintas:

- *PerformerAgent Interface*: esta interface permite que tarefas sejam delegadas a seus executores e que tarefas sejam pesquisadas e controladas (suspensas, reiniciadas ou abortadas);
- *SourceAgent Interface*: permite que executores devolvam o resultado das tarefas a quem as requisitou (requisitante);
- *Worklist Interface*: *worklist* é uma lista de trabalho, a partir da qual executores tomam conhecimento das tarefas que devem realizar no sistema de workflow. Através desta lista, participantes têm acesso às requisições de tarefas enviadas a cada executor pelos vários requisitantes.

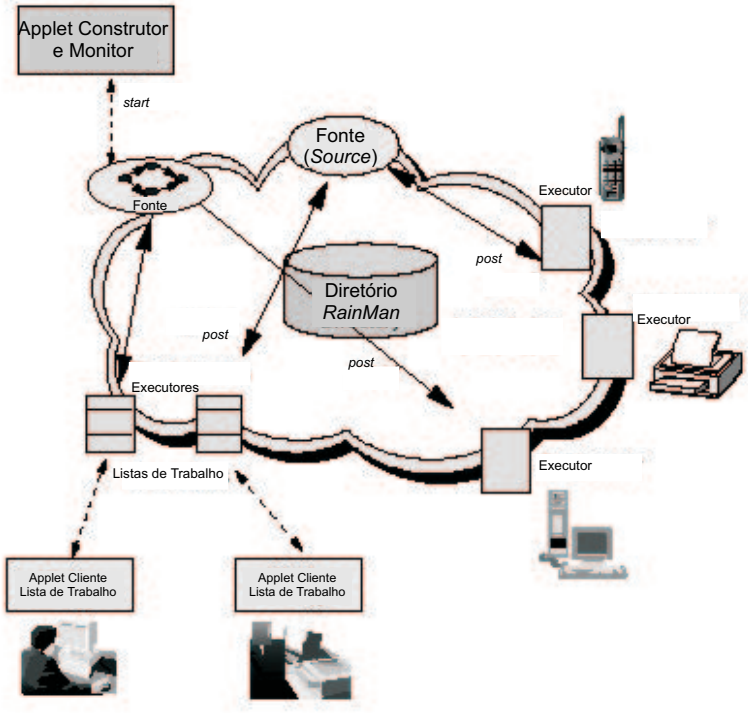

Figura 13: Arquitetura RainMan [13].

Dessa forma, o sistema RainMan é montado a partir destas interfaces. A arquitetura do sistema é apresentada na Figura 13.

Conforme mostra a figura, os requisitantes de serviços, através de uma chamada *post*, requisitam aos executores a execução de determinadas atividades. Cada executor, por sua vez, mantém uma lista das tarefas que deve executar. No diretório RainMan os requisitantes têm a possibilidade de buscar pelos executores que têm a capacidade de executar determinadas tarefas.

Usuários do workflow interagem com o ambiente de execução em RainMan através dos chamados componentes de interface de usuário. Atualmente são três componentes, conforme descrito em [13]:

- Applet RainMan Construtor;
- Applet Cliente de Lista de Trabalho RainMan;
- Interface de Administrador RainMan.

O primeiro dos componentes, o Applet RainMan Construtor, é um ambiente gráfico de especificação de workflows, que atua como interpretador do grafo de workflow e no monitoramento da execução de uma instância do workflow. Neste ambiente, workflows são especificados como grafos acíclicos direcionados.

Quando um workflow está sendo executado, este ambiente entrega as requisições de tarefas para os executores específicos, sucessivamente a cada tarefa efetuada. Além disso,

o Applet RainMan Construtor interpreta o grafo do workflow dinamicamente, de modo que qualquer alteração em tempo de execução pode ser efetuada. Esse é um dos grandes diferenciais de RainMan para os demais sistemas de workflow. Essa característica é muito importante também pelo fato de que a autonomia dos executores permite que exceções sejam geradas quando da execução de uma tarefa requisitada, além de que as suas capacidades podem mudar no tempo, assim como seu estado de conexão com a rede, em especial quando se trata de ambiente de rede móvel. Apesar de Construtor, Requisitante (*Source*) e Monitor serem entidades lógicas distintas em RainMan, elas são parte do mesmo applet Construtor.

O applet que gerencia a lista de tarefas de cada usuário executor é o Applet Cliente de Lista de Trabalho RainMan. Através dele, cada usuário tem acesso à sua lista de trabalho a partir de um browser Web. Usuários também podem ver esta lista remotamente e realizar download de tarefas específicas.

Task Handlers são iniciados automaticamente a cada requisição enviada do requisitante para um executor, de acordo com a necessidade de cada requisição e com a capacidade do seu executor. Por exemplo, se uma tarefa é processar um documento e o seu executor tem essa capacidade, o *Task Handler* efetua a chamada à aplicação de processamento de documento. Após o término do processamento, o *Task Handler* retorna a resposta da execução para a sua lista de trabalho, que por sua vez a envia para o requisitante.

Uma vez que cada requisição de tarefa da lista de trabalho é executada localmente na máquina do executor, operação em modo desconectado é permitida. O applet do cliente necessita apenas se conectar à sua lista de trabalho remota para o envio e o recebimento de tarefas. No entanto, se dados ou aplicações adicionais são necessários para a execução de uma tarefa, uma conexão deve ser estabelecida entre o cliente e a rede. O applet do cliente possui um significado importante no sistema, pelo fato de que ele demonstra como executores podem receber tarefas de fontes de workflows distintas a partir de uma única interface.

Além dos applets já citados, um outro componente importante do sistema RainMan é o seu ambiente de execução, que fornece um conjunto de serviços distribuídos, os quais são executados na Internet. O serviço de diretórios, por exemplo, fornece aos requisitantes de tarefas a possibilidade de identificar os executores que são capazes de efetuar determinada tarefa. Desta forma, tarefas podem ser alocadas a executores dinamicamente, uma vez que estes mantém seus registros neste diretório. Para usuários móveis que executam tarefas, o serviço de diretório é utilizado para encontrar suas listas de tarefas. Esse tipo de serviço é o que permite que executores sejam também usuários em dispositivos móveis.

Diretórios são organizados hierarquicamente por domínio administrativo, e requisitantes podem encontrar executores em qualquer domínio. Cada executor é registrado em seu domínio local e, caso um requisitante não seja daquele mesmo domínio, ele pode encontrar o executor através do serviço de diretório de seu domínio, o qual faz uma busca hierárquica nos diretórios.

Um outro serviço disponível em RainMan é o serviço de listas de trabalho, as quais são tratadas como objetos de rede endereçáveis. Deste modo, a troca de mensagens (requisições e respostas a elas) é feita de forma assíncrona. Este serviço permite que requisições possam ser repassadas a executores mesmo que estes não estejam conectados

(estas requisições são colocadas em sua lista de trabalho). Além disso, listas de trabalho podem ser compartilhadas entre diversos requisitantes. Tarefas são, portanto, repositórios intermediários, acessíveis aos executores e requisitantes sempre que estes estão conectados à rede.

Com relação à gerência de falhas, RainMan possui certas dificuldades, uma vez que é um ambiente descentralizado de workflow. No entanto, a interação entre o requisitante e o executor pode ser rica o suficiente para manipular as falhas e as possíveis alternativas diante de situações inesperadas. Por exemplo, se um requisitante é um workflow de agendamento de viagens e um dos executores um servidor de serviço de hotel, pode ocorrer do primeiro agendar uma viagem e depois resolver cancelá-la. Neste caso, a operação de cancelamento disponibilizada pelo hotel pode ser vista como uma operação de compensação para a operação de agendamento.

Quanto à execução descentralizada, RainMan separa claramente requisitantes, responsáveis pela coordenação de cada processo, de executores espalhados pela rede, responsáveis pela execução de tarefas do processo. Desta forma, requisitantes diversos podem compartilhar executores autônomos.

Dessa forma, RainMan é um sistema de workflow descentralizado, distribuído via Internet, que permite que workflows sejam atualizados durante a execução e que usuários móveis no papel de executores possam operar em modo desconectado. Atualmente são 60 classes Java desenvolvidas em Java JDK 1.1.4, utilizando RMI (*Remote Method Invocation*). As próximas versões têm por objetivo separar um pouco das funções desempenhadas pelo Applet Construtor, implantar um mecanismo efetivo de prioridade e deadline de tarefas e aumentar o número de servidores de listas de tarefas, a medida que for crescendo o número de usuários executores. O artigo não apresenta questões de controle de concorrência e detecção e resolução de conflitos.

5.4 Arquitetura de Workflow para Gerência de Trabalho Móvel Colaborativo

Em [8] os autores propõem uma arquitetura de workflow para gerência de trabalho colaborativo em ambiente móvel, assumindo que os usuários móveis estão em constante conexão com o sistema, o qual pode ser visto como uma hierarquia de workflows descentralizados com diferentes níveis de especificação e granularidade de controle de tarefas.

Os autores de [8] explicam que a abordagem de sistemas de gerência de workflow convencional, como aquela proposta pela WfMC, não suporta a execução de atividades colaborativas em um ambiente descentralizado integrado a uma rede de computação móvel. Isso porque WfMC implementa uma tecnologia rígida de modelos hierárquicos de coordenação de tarefas, através da qual a administração do sistema impõe total autoridade sobre a execução de processos, já que o módulo de execução é implementado como um servidor central.

No sistema proposto em [8], são distinguidos três papéis, a citar: administradores, que acessam centralizadamente o estado corrente da execução do workflow; projetistas, que são os donos dos workflows; e equipes, responsáveis pela execução das tarefas dos workflows. Administradores e projetistas têm acesso ao sistema de gerência de workflow, o qual é similar a um sistema cliente/servidor centralizado tradicional.

O núcleo do sistema é apoiado por uma infra-estrutura estacionária, baseada em uma rede fixa de alta performance. Os administradores sempre utilizam os computadores estacionários. Os projetistas e as equipes de execução, por sua vez, muitas vezes se utilizam de dispositivos móveis para o trabalho em campo. Para estas equipes, estes dispositivos são necessários principalmente quando precisam acessar o banco de dados do sistema. A conexão das equipes à rede é considerada intermitente.

As operações básicas de um sistema de gerência de workflow são providas pelo núcleo do sistema, que armazena informações relativas ao estado de execução de cada workflow e demais informações relevantes: projetista/dono do workflow, definição e agenda das tarefas, estado corrente de cada tarefa, dentre outras. Algumas informações do tipo comentários e anotações são consideradas *append-only information* e, por isso, podem ser atualizadas concorrentemente, sem gerar conflitos posteriores.

Canais de eventos

Nesse sistema, existe um canal de eventos N-para-N associado a cada workflow, através do qual são propagados os eventos relativos às transições de estado do workflow e suas tarefas. A este canal de evento têm acesso o projetista do workflow e todas as equipes de execução. A cada projetista também existe associado um outro canal N-para-N, por meio do qual ele anuncia as tarefas que ainda não foram designadas a nenhuma equipe. A partir destes canais, projetistas e equipes de trabalho recebem e enviam eventos, os quais são todos armazenados pelo núcleo do sistema de modo persistente. Os dois canais podem também propagar eventos que não estão diretamente relacionados à atualização dos dados, caracterizados como eventos de percepção.

Esses canais de evento também suportam comunicação multicast, como em uma lista de e-mail. No entanto, neste caso a persistência somente é garantida aos eventos que chegam ao núcleo do sistema. Por outro lado, todos os eventos enviados pelo núcleo do sistema são persistentes e ordenados no tempo. Quando dispositivos móveis enviam pelo canal eventos que se referem à uma atualização local, estes eventos são novamente espalhados via multicast quando o núcleo envia um evento indicando que essa atualização se tornou estável. Estas duplicações de informação no canal são facilmente detectadas.

Quando um grupo de trabalho precisa executar uma tarefa T_2 que depende da execução de uma outra tarefa T_1, os canais de eventos são utilizados para a obtenção da informação associada à conclusão desta tarefa T_1. Se o grupo que propagou a informação de que T_1 foi terminada mantinha o bloqueio sobre aquela tarefa, então a atualização por ela realizada será propagada e se tornará estável no núcleo do sistema. Dessa forma, a informação de que o primeiro grupo necessita está disponível no canal de eventos e pode ser por ele utilizada para a execução da tarefa T_2.

Gerência de Cache e Trabalho Cooperativo Desconectado

Os dispositivos móveis podem armazenar em cache informações relativas ao status de execução do workflow, ao status de cada equipe, às tarefas dos workflows, dentre outras. Enquanto os projetistas normalmente têm acesso a toda informação relativa ao workflow e às equipes de trabalho envolvidas em sua execução, estas equipes apenas armazenam as

tarefas que lhes competem e algumas que não foram delegadas a nenhuma equipe, mas que podem ser por elas executadas. Dessa forma, o projetista normalmente armazena uma maior quantidade de informação. Para ajudar as equipes a coordenar suas atividades, que podem fazer parte de mais de um workflow, os dispositivos móveis possuem uma aplicação local autônoma para a gerência de suas listas de tarefas. O workflow e o próprio mecanismo de gerência de tarefas fornecem dicas sobre quais objetos devem ser colocados no cache dos dispositivos móveis dos projetistas e das equipes de trabalho.

O núcleo do sistema mantém a informação atualizada de todos os workflows. Para que uma atualização torne-se válida, ela deve ser primeiro aplicada e tornada estável no núcleo do sistema. Desta forma, a política de controle de concorrência, que será descrita a seguir, é baseada em um mecanismo de bloqueio que permite que usuários móveis realizem alterações sobre as informações relativas aos workflows que executam se obtiverem anteriormente o bloqueio do item de dado correspondente. Informações que não geram conflitos, que são apenas informações adicionais (*append-only*), não necessitam ser bloqueados. As atualizações são primeiro aplicadas ao cache local de cada usuário e posteriormente propagadas e definitivamente validadas no núcleo do sistema.

O status dos diferentes workflows é mantido pelo núcleo do sistema. Este status somente pode ser atualizado quando transações de atualização são executadas no núcleo. Pelas características dos canais de eventos, uma atualização realizada durante um período de desconexão pode ser armazenada pelo núcleo do sistema e vista por outras equipes de trabalho, mas posteriormente pode ser abortada devido ao fato de violar os requisitos da política de controle de concorrência quando ela finalmente alcançar de maneira segura o núcleo.

Além de objetos em cache e das atualizações locais, dispositivos móveis podem também armazenar as atualizações recebidas das equipes de trabalho remotas através dos canais de eventos. O cache destes dispositivos implementa uma política de versões. Sempre que possível, para cada objeto no cache são mantidas duas versões: uma delas é a versão correspondente àquela armazenada no núcleo do sistema e, portanto, consistente. A outra versão se refere a um valor local, que apenas pode se tornar persistente e estável se propagada para o núcleo do sistema. Quando o dispositivo móvel se conecta ao núcleo para a troca de informação, estas duas versões dos objetos em cache podem se tornar similares. No momento da reconexão, o dispositivo móvel informa ao núcleo suas atualizações locais, que tornam-se então persistentes no núcleo. Neste momento, a versão tentativa passa a ter o mesmo valor da versão persistente no cache do dispositivo móvel. O gerenciamento de versões tentativas é opcional.

De acordo com a política de controle de concorrência implantada por esse sistema, atualizações locais são efetuadas somente sobre objetos sobre os quais se tem bloqueio e, exceto em situações raras, nenhuma atualização remota é recebida sobre os mesmos objetos.

Delegação de Tarefas e a Política de Controle de Concorrência

No sistema de workflow que está sendo apresentado, existem dois mecanismos para associar tarefas a equipes de trabalho. Equipes podem consultar o sistema sobre as tarefas que devem executar (modelo fechado de distribuição de tarefas) ou o sistema pode enviar

para as equipes as tarefas a serem executadas, através dos canais de eventos (modelo aberto). Em determinadas situações, tarefas podem ser reassociadas a outras equipes.

Para o modo aberto existem diversas maneiras de se eleger a equipe a executar determinada tarefa, como segue:

- a equipe a executar uma tarefa pode ser a primeira equipe que efetuou uma transação no núcleo do sistema para executar esta tarefa;
- a equipe a executar uma tarefa pode ser escolhida de acordo com uma política de concordância entre as partes envolvidas.

Nestes dois casos, o projetista tem o poder da decisão final, uma vez todas as tarefas ainda não delegadas a equipes estão por ele bloqueadas.

De acordo com o mecanismo de controle de concorrência, quando uma tarefa é designada à execução por uma equipe, ela permanece bloqueada por esta equipe. No entanto, este bloqueio é considerado revogável e deve ser periodicamente reassociado às equipes. Enquanto uma equipe mantém o bloqueio da tarefa que executa, o sucesso das operações que realiza é garantido. Se a equipe perde o bloqueio, a tarefa passa a ser bloqueada pelo projetista. As operações realizadas pela equipe que mantinha o bloqueio são armazenadas no núcleo do sistema na forma de tentativas e podem ser reutilizadas pelos projetistas para reintegrar a informação relevante associada àquela tarefa. O período que uma equipe tem para reestabelecer o bloqueio da tarefa é variável e depende do tempo necessário para a execução da tarefa, o qual é controlado pelo sistema.

Um projetista pode, quando detém o bloqueio sobre uma tarefa, associá-la a outra equipe de trabalho ou à mesma equipe que perdeu o seu bloqueio. Além disso, a qualquer momento, como mantém conexão ininterrupta com a rede, um projetista pode reassociar ou desassociar uma equipe de uma tarefa. Neste caso, um evento é imediatamente propagado pelo canal para que os bloqueios correspondentes sejam revogados.

5.5 Modelo CoAct

No contexto do projeto TRANSCOOP [11], do *German National Center for Information Technology* (GMD), foi desenvolvido um modelo de transação cooperativo chamado *Cooperative Activity Model* (CoAct), o qual oferece um framework para atividades cooperativas de longa duração em ambientes multi-usuários e interativos.

Em CoAct, o mecanismo de replicação otimista define que cada usuário membro do grupo em colaboração tem seu próprio espaço de dados, denominado de espaço de trabalho privado (*private workspace*). Neste espaço de trabalho cópias de ítens de dados são mantidas para que o usuário possa executar suas tarefas independentemente dos demais. Nele é armazenado um log com as modificações realizadas pelo usuário sobre os ítens de dados que mantém cópia. Este log é chamado de histórico do espaço de trabalho, que é formado por uma seqüência de entradas de história totalmente ordenadas. Cada entrada consiste de um identificador de operação, que é único com relação à toda a atividade cooperativa, e da descrição da operação (nome, valores de parâmetros de entrada e de saída) invocada pelo usuário.

Além do espaço de trabalho privado, existe em CoAct um espaço de trabalho comum para cada atividade cooperativa. Este espaço de trabalho é isolado dos espaços de dados

de cada usuário e cada um destes pode fazer acesso a ele. Este espaço comum representa, no final da execução da atividade cooperativa, o resultado da cooperação. A partir deste espaço são feitas as cópias de dados para os espaços privados de cada usuário.

Cada atividade cooperativa (workflow) é descrita por:

- um conjunto de operações (que representam tarefas atômicas) que podem ser executadas por um usuário sobre o seu espaço de trabalho privado. Cada operação é considerada atômica e sempre leva o espaço de trabalho de um estado consistente para outro estado consistente. A seqüência de execução das operações é definida pelo usuário em tempo de execução. Uma operação pode ser, por exemplo, a escrita do valor de uma variável sobre o seu espaço de trabalho;

- um conjunto de regras de *merging* que definem, através da semântica das operações, o processo de troca de informação, chamado de *merging history*. Esta troca de informação ocorre para que usuários possam informar para os demais ou para o espaço de trabalho comum as operações por ele executadas ou mesmo receber as operações executadas por outros usuários.

A Figura 14 apresenta a maneira com que os usuários de CoAct trocam informação e colaboram entre si para alcançar o objetivo final da atividade cooperativa.

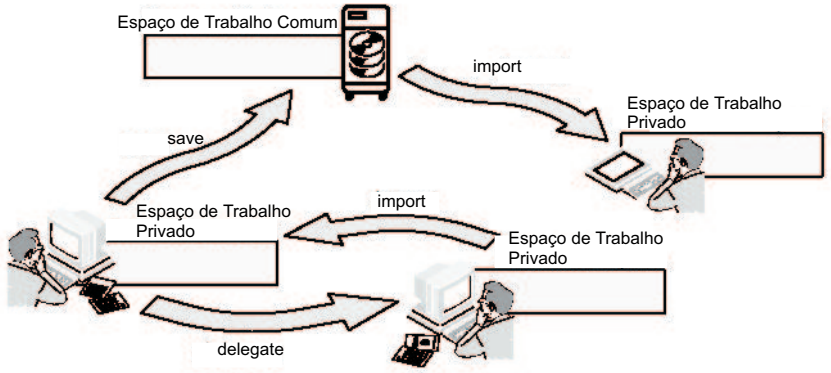

Figura 14: Troca de informação entre espaços de trabalho em CoAct [11].

Em CoAct, operações são a unidade de troca de informação, e não os dados em si. Elas representam a menor unidade de trabalho em uma atividade cooperativa. O modelo de CoAct fornece duas opções para a troca de informação:

- troca de informação direta entre espaços de trabalho;

- troca de informação entre o espaço de trabalho privado e o espaço comum.

Na primeira opção, a troca de informação ocorre através das chamadas *import* e *delegate*. A chamada *import* permite que um membro do grupo em cooperação incorpore em seu próprio espaço de trabalho operações executadas por outro usuário em seu espaço. O usuário torna-se, no momento da chamada, o responsável pela detecção e resolução de possíveis conflitos.

A chamada *delegate*, por sua vez, possibilita que o membro em cooperação transmita a outro o conjunto de operações que realizou sobre seu espaço de dados. O receptor, neste caso, é o responsável pela resolução de conflitos.

Na segunda opção, a troca de informação com o espaço de trabalho comum pode ser feita por meio das chamadas *save* e *import*, conforme apresentado na Figura 14. Através de *save* o usuário em cooperação pode tornar disponível para os demais no espaço de trabalho comum o resultado de sua computação. O usuário que executa a chamada é o responsável pela resolução de conflitos. Esta informação colocada neste espaço pode ser então recuperada pelos demais membros em cooperação através da chamada *import*.

Trabalho Cooperativo Desconectado

Para que fosse possível a operação de usuários na atividade cooperativa por meio de dispositivos móveis, CoAct estendeu sua arquitetura, conforme apresentado na Figura 15.

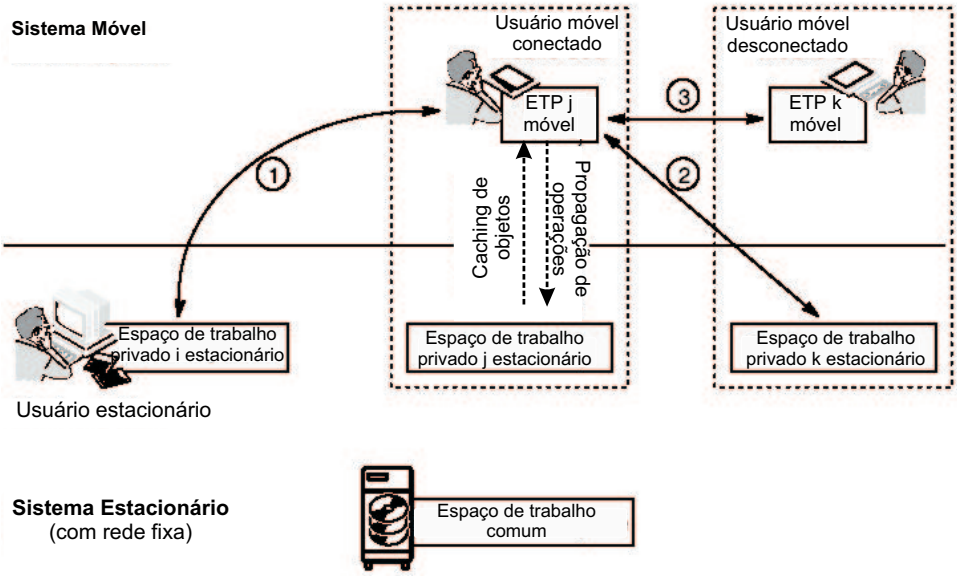

Figura 15: Arquitetura estendida de CoAct [11].

Nesta arquitetura entendida, além de cada usuário ter seu espaço de trabalho privado (ETP) na rede fixa, ele tem um espaço de trabalho no seu dispositivo móvel conectado à rede sem fio. Este espaço de trabalho móvel serve como um cache de dados. Algumas vantagens em se ter um espaço de trabalho por usuário para cada uma das redes, fixa e móvel, podem ser citadas:

- apesar do espaço de trabalho móvel servir como cache, como o espaço de armazenamento do dispositivo móvel é normalmente pequeno, nem todos os dados podem estar nele ser colocados;

- como dispositivos móveis são mais suscetíveis a estragos, os dados nele armazenados não são duráveis. Daí a necessidade de se armazenar de forma persistente e durável as atualizações realizadas pelo usuário e todo o histórico do seu espaço de trabalho;

- mesmo quando desconectado da rede, as informações contidas no espaço de trabalho de um usuário podem ser recuperadas através de seu espaço de trabalho na rede fixa.

A forma de cooperação de um usuário na rede fixa permanece inalterada na arquitetura estendida. No entanto, os usuários em dispositivos móveis executam suas operações em seus espaços de trabalho móveis. Os dados necessários para a execução de cada operação devem ser previamente trazidos para este espaço. Se uma operação necessitar de um dado que não esteja neste espaço móvel, o *cache miss* pode ser resolvido imediatamente se o usuário estiver conectado à rede. Neste caso, ele precisa apenas buscar pelos ítens de dados no espaço de trabalho estacionário.

Como as operações de um usuário móvel são executadas no espaço de trabalho móvel de seu dispositivo de acesso, essas operações devem ser propagadas assim que possível para o espaço de trabalho fixo, a fim de que informações não sejam perdidas e que este espaço na rede fixa seja mantido sempre atualizado. Uma vez recebidas as operações, elas devem ser executadas no espaço de trabalho estacionário, para que o estado de ambos os espaços seja consistente. Se o usuário encontra-se desconectado da rede, este processo deve ser efetuado logo após reconexão. No espaço de trabalho da rede fixa todas as entradas no histórico são mantidas até o fim da atividade cooperativa.

No que diz respeito à troca de informação, esta arquitetura estendida permite que a informação seja trocada entre:

- o espaço de trabalho estacionário de um usuário da rede fixa e o espaço de trabalho móvel de outro usuário (número 1 na Figura 15);
- o espaço de trabalho móvel de um usuário e o espaço de trabalho estacionário de um usuário móvel (número 2 na Figura 15);
- espaços de trabalho móveis de dois usuários na rede móvel (número 3 na Figura 15). Para isso, cada usuário deve estar conectado à rede fixa ou já ter trazido para seu espaço de trabalho móvel todo o seu histórico de atualizações.

Gerência de Cache

A fim de minimizar o número de *cache misses*, devem ser trazidos para o cache do usuário móvel os dados que serão necessários para a execução de suas operações. Assim como no sistema de arquivos Coda [10], o conteúdo do cache é determinado de acordo com:

- ítens de dados que estão sendo correntemente usados pelo usuário para a execução das tarefas;
- ítens de dados que provavelmente serão necessários ao usuário para a realização de operações futuras.

Ítens de dados podem ser removidos do cache de um usuário mesmo se já modificados, caso ainda não tenham sido enviados de volta para o espaço de trabalho comum.

O sistema de autoria hipermídia SEPIA é uma aplicação demonstrativa do sistema CoAct. Em SEPIA, é definida uma métrica de relevância para os dados, de modo que, no caso de ocorrência de *cache miss*, não somente são trazidos para cache os ítens de dados

requisitados, mas também todos aqueles que possuem uma relevância para o item de dado requisitado, de acordo com um valor pré-definido.

Uma vez que os dados tenham sido trazidos para cache, um usuário em um dispositivo móvel pode operar em modo desconectado. No entanto, dois problemas são levantados em [11]: a gerência de cache quando o usuário está operando em modo desconectado e a manutenção de identificadores de operação e de ítens de dados únicos dentro da atividade cooperativa. A resolução de conflitos, por outro lado, é uma característica inerente do sistema, já que ocorre no momento do *history merging*.

Em CoAct *cache misses* são manipulados de forma a não bloquear a operação em execução pelo usuário. Se não for possível ou caso o usuário não queira se conectar à rede fixa para buscar informação em seu espaço de trabalho nesta rede, a execução da operação é abortada e o usuário é informado da ocorrência do *cache miss*. Se o usuário se conecta à rede, a busca do item de dados não encontrado no espaço de trabalho móvel é transparente para o usuário.

O problema de manutenção de identificadores únicos para operações e ítens de dados, no entanto, não ocorria na arquitetura convencional de CoAct, onde não eram previstos usuários em dispositivos móveis, já que a unicidade dos identificadores era garantida pela instância central que gerenciava o espaço de trabalho comum.

Para que essa unicidade seja garantida em caso de desconexão, a cada par de espaço de trabalho estacionário e espaço de trabalho móvel é associado um identificador de espaço de trabalho único. Este identificador é dinamicamente gerado pelo espaço de trabalho comum na rede fixa e associado aos espaços de trabalho de cada usuário quando este se junta ao grupo de trabalho cooperativo para a execução de uma atividade. Além deste identificador, cada par de espaços de trabalho mantém um contador de operações local. Para que a unicidade seja garantida, dessa forma, o identificador final de uma operação consiste do identificador do espaço de trabalho e do valor corrente do contador de operação local.

Para o caso de objetos (ítens de dados), a abordagem de CoAct é semelhante. Cópias do mesmo item em diferentes espaços de trabalho possuem o mesmo identificador. Para evitar que objetos criados em diferentes espaços de trabalho privados possuam o mesmo identificador, assim como ocorre com operações, o identificador de um objeto em CoAct consiste de identificador do espaço de trabalho e do valor do contador local de objetos.

Reintegração

Como em CoAct a informação trocada e propagada entre usuários distintos, entre espaços de trabalho de um mesmo usuário e entre usuários e o espaço de trabalho comum é a operação e não o item de dado modificado, a reintegração consiste da re-execução das operações no espaço de trabalho na rede fixa. Essa troca de informação reduz a largura de banda necessária e pode ser feita assincronamente. Como não ocorre *merge*, a reintegração não gera conflitos. As entradas no espaço de dados móvel apenas são acrescidas às entradas do espaço de dados na rede fixa.

Caso a propagação das operações do espaço de trabalho móvel para o espaço de trabalho estacionário do mesmo usuário não seja possível, devido a falhas na rede fixa ou porque a máquina onde se encontra o espaço de trabalho privado está fora do ar, o modelo

CoAct permite que as operações tornem-se visíveis aos demais usuários em cooperação antes mesmo de serem propagadas para o espaço de trabalho na rede fixa.

Otimizações são possíveis para reduzir o tamanho do histórico do espaço de trabalho móvel a ser transferido. Se uma operação é anulada por uma operação posterior, a sua entrada pode ser removida do histórico de atualizações.

Recuperação de Falhas

Falhas em ambientes móveis são bem mais freqüentes do que falhas em ambientes de rede fixa. Elas podem ser classificadas em duas categorias: falhas permanentes, as quais não podem ser tratadas autonomamente pelo dispositivo móvel, e falhas transientes, as quais afetam temporariamente o funcionamento do dispositivo.

Falhas transientes em CoAct são recuperadas através de conceitos de recuperação de banco de dados tradicionais. Antes de uma operação ser executada, a sua entrada no histórico é escrita em meio persistente. Em caso de falha transiente, a último estado do espaço de trabalho da rede fixa pode ser recuperado pela re-execução das operações armazenadas, o que pode ocorrer periodicamente.

Falhas permanentes, por outro lado, são manipuladas em CoAct através do uso do espaço de trabalho na rede fixa como um backup confiável do espaço de trabalho móvel. Operações executadas no espaço de trabalho móvel durante desconexão são imediatamente propagadas para o espaço de trabalho estacionário quando possível.

5.6 WorkToDo

WorkToDo [16] é um sistema de gerenciamento de workflows para ambientes de comunicação sem fio desenvolvido na Universidade Estadual de Campinas (Unicamp). Nele, que usuários em cooperação podem executar tarefas no modo desconectado. Quando há conexão, esta é considerada instável e de baixa largura de banda. Este sistema utiliza o conceito de reserva de tarefas e transferência antecipada de dados e aplicações para o dispositivo móvel, assim como o faz Exotica e o sistema que utiliza canais de eventos para a comunicação.

Em WorkToDo, um processo é um conjunto de atividades (tarefas ou subprocessos) e as dependências entre elas, que determinam as condições em que elas podem ser executadas. Cada atividade é formada por um conjunto de dados de entrada e saída e por uma entidade processadora, cuja responsabilidade é executá-la. Uma entidade processadora pode ser um usuário ou uma aplicação. Uma tarefa, por sua vez, é a menor unidade de execução e não pode ser decomposta. Os dados à ela relativos são especificados na definição do processo.

Cada processo em execução em WorkToDo utiliza uma área exclusiva para armazenamento de dados, chamada de Contexto do Processo. Operações de recuperação e atualização de dados pelas tarefas são realizadas nesta área. O contexto de uma instância de um processo somente é acessível às tarefas daquela instância.

A arquitetura distribuída de WorkToDo é apresentada na Figura 16. Métodos de comunicação remota tais como RPC (*Remota Procedure Call*), CORBA (*Common Object Request Broker Architecture*) e RMI (*Remote Method Invocation*) devem ser utilizados

para permitir a comunicação entre os diversos componentes que rodam em máquinas distintas distribuídas pela rede.

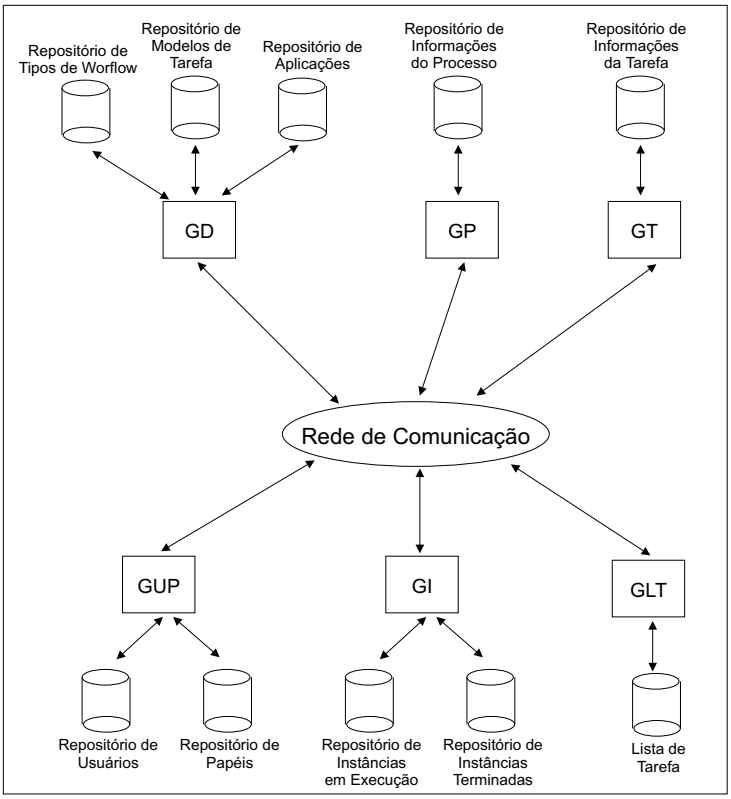

Figura 16: Arquitetura WorkToDo [16].

Conforme apresentado na figura, são seis os componentes de WorkToDo:

- Gerenciador de Usuários e Papéis (GUP): o responsável por gerenciar usuários e papéis, adicionando-os, removendo-os, consultando-os nos repositórios correspondentes;
- Gerenciador de Definições (GD): o responsável por gerenciar as definições dos processos, tarefas e aplicações, armazenadas nos repositórios correspondentes;
- Gerenciador de Instâncias (GI): o responsável por manter um registro das instâncias de processo em execução e daquelas já encerradas, cada registro em seu repositório correspondente (repositório de instâncias em execução e repositório de instâncias terminadas). Ele também tem como tarefa criar um Gerenciador de Processo (GP) para cada instância, para o qual ele transfere, no momento da criação, uma cópia da definição do processo. Para tratar possíveis falhas dos GPs, o GI armazena no repositório de instâncias de processos em execução o estado atual da execução de cada instância. Se a falha ocorrer, o GI cria um novo GP em uma outra máquina com o último estado consistente do GP que falhou;
- Gerenciador de Processo (já citado GP): o responsável por coordenar a execução de um processo. Ele verifica quais tarefas estão prontas para serem executadas e

inicia sua execução. Os resultados coletados são analisados para se encontrar as próximas tarefas a serem executadas. Este gerenciador armazena, além da lista de tarefas que compõem o processo, o estado da instância do processo. São dois os seus subcomponentes:

  - o escalonador, que avalia as condições de execução das tarefas, determinando quais tarefas podem ser executadas em um determinado momento;
  - o despachante, que prepara as tarefas para execução. Se a tarefa é automática, ou seja, é executada por uma aplicação sem o auxílio do usuário, o despachante cria um Gerenciador de Tarefa (GT) para controlar sua execução. Caso contrário, de acordo com uma Política de Notificação escolhida no momento da criação da instância e modificável durante sua execução, o despachante notifica os usuários pertencentes ao papel da tarefa a respeito de sua disponibilidade.

  O GP também deve tratar de eventuais falhas nos Gerenciadores de Tarefas. Caso elas ocorram, o GP pode criar um novo GT com o último estado conhecido do GT que falhou. Além disso, o Gerenciador de Processo é o responsável por controlar o prazo das tarefas, já que cada tarefa em WorkToDo pode ter um prazo de execução, especificado em sua definição. No entanto, quando o usuário está inativo e, portanto, sem poder receber notificação do GP a respeito do prazo da tarefa, o Gerenciador de Lista de Trabalho passa a ter o controle sobre os prazos das tarefas nesta lista, notificando o usuário. Se o prazo de uma tarefa é ultrapassado, o responsável pelo processo pode decidir se a execução do processo deve ou não ser interrompida;

- Gerenciador de Tarefas (já chamado de GT): o responsável por controlar a execução de uma tarefa automática. Ele é criado pelo GP com a definição da tarefa. A partir desta, ele invoca a aplicação correspondente para a sua execução. Quando a execução da tarefa é completada, o Gerenciador de Tarefa notifica o Gerenciador de Processo que o criou, informando se a execução terminou com sucesso ou falha. Caso a execução tenha falhado, ele pode executar novamente a tarefa, até que ela seja completada com sucesso ou até que um número pré-definido de tentativas em sua definição seja alcançado;

- Gerenciador de Lista de Trabalho (GLT): o responsável por monitorar a lista com as tarefas que um determinado usuário pode executar, denominada Lista de Trabalho. Nesta lista, cada tarefa é chamada de item de trabalho. Tarefas automáticas, uma vez que são executadas automaticamente pelo sistema, não são colocadas nesta lista. Para cada usuário sempre existe apenas um GLT.

A interação do usuário com o sistema WorkToDo ocorre principalmente através de sua lista de trabalho. Quando o usuário deseja executar algum dos ítens relacionados em sua lista, ele deve selecioná-lo. Quando selecionado, um item de trabalho é removido pelo Gerenciador de Processo da lista de trabalho dos demais usuários que poderiam executá-lo, evitando que mais do que um usuário execute um mesmo item. O usuário é então avisado e pode começar a execução.

Caso o usuário deseje executar um determinado item de trabalho enquanto desconectado (usuário considerado inativo), ele deve bloqueá-lo. Assim como quando o usuário

vai executar um item em modo conectado, o item é retirado da lista dos demais usuários. Um item somente pode ser bloqueado se a tarefa a ele correspondente foi definida como sendo passível de ser realizada no modo desconectado.

Além de poder realizar diversas operações em sua lista de trabalho (consulta, seleção e bloqueio de ítens), um usuário pode verificar quais processos estão em execução e consultar o estado de execução daqueles dos quais participa.

Além de usuários executores, existem em WorkToDo dois outros tipos de usuários:

- usuário administrador: aquele que consulta e altera informações relativas a usuários, papéis, tipos de processos, tarefas e aplicações;
- usuário responsável pelo processo: aquele que toma as decisões com relação ao processo, sempre que necessário.

Prefetching e Trabalho Cooperativo Desconectado

Em WorkToDo a técnica de *prefetching* é utilizada para copiar para a máquina do usuário ítens de trabalho que podem vir a ser executados mais tarde quando o usuário estiver desconectado do sistema. Quando estes ítens forem bloqueados, eles já podem ser executados porque as informações necessárias para execução já estarão disponíveis em cache. Quando se reconecta ao sistema, o usuário deve informar os resultados dos ítens executados.

Essa técnica de *prefetching* é executada em WorkToDo pelo Gerenciador de Lista de Trabalho, que durante conexão verifica quais os ítens da lista que podem ser executados em modo desconectado e os copia para a máquina do usuário, com todas as informações associadas necessárias para execução. Quando toda a cópia de um item e seus dados é efetuada, este item é considerado transferido. Isso ocorre com cada um dos ítens da lista de trabalho do usuário, conforme estão nela ordenados. No entanto, ítens já bloqueados possuem prioridade sobre os demais. Uma desvantagem desta técnica é que, caso o usuário não bloqueie um item, seus dados apenas estarão ocupando o espaço de armazenamento (sempre limitado) de seu dispositivo.

Quando a desconexão não é planejada, ocorrendo a qualquer momento durante o trabalho do usuário, o Gerenciador de Lista de Trabalho deixa habilitados para execução apenas os ítens de trabalho que estavam bloqueados pelo usuário e que já tinham sido transferidos para a sua máquina. Se, no momento da desconexão, o usuário tinha algum item selecionado, podem ser considerados dois casos:

- a execução da tarefa é interrompida porque este item de trabalho utiliza dados que não estão na máquina do usuário. Neste caso, quando o usuário se reconectar ele poderá voltar à execução deste item porque neste momento ele tem novamente acesso aos dados;
- a execução da tarefa é terminada porque esta tarefa utiliza uma aplicação que não está instalada na máquina do usuário. Neste caso, quando o usuário se reconectar ele deverá reiniciar a aplicação para a retomada da execução da tarefa.

A subseção seguinte apresenta uma comparação entre os sistemas citados.

5.7 Comparação entre os Sistemas de Workflow

Entre os sistemas apresentados, existem diversas semelhanças e diferenças, as quais serão aqui expostas.

Magi, Exotica, RainMan e os outros sistemas apresentados são todos sistemas de workflow que permitem o uso por usuários em dispositivos móveis. No entanto, em Magi não é possível operar em modo desconectado.

Com relação à distribuição de tarefas, na maioria dos sistemas os usuários recebem as tarefas que devem executar. Apenas no sistema que utiliza o conceito de canais de eventos [8] os usuários podem tanto recebê-las quanto consultá-las através do sistema.

Com relação ao trabalho em modo desconectado, a maioria dos sistemas emprega uma política pessimista de controle de concorrência, através do conceito de bloqueio de tarefas. Para que uma tarefa seja executada no modo desconectado nestes sistemas, ela deve ser bloqueada e toda a informação necessária para a sua execução, incluindo os artefatos/recursos associados, deve ser previamente replicada no dispositivo móvel do usuário executor.

No que se refere à cooperação, em todos os sistemas estudados os usuários interagem entre si por meio de mensagens trocadas ou através de áreas para troca (leitura e escrita) de informação.

Com relação à coordenação, todos os sistemas, exceto WorkToDo, possuem um ponto central de coordenação. Em WorkToDo são responsáveis pela coordenação os diversos componentes gerenciadores.

No que diz respeito à resolução de conflitos, em CoAct ela é toda resolvida pelo usuário que está entregando a informação ao espaço de dados comum ou pelo usuário que está recebendo de outra o seu conjunto de operações.

Uma questão interessante em Magi é o fato dos serviços serem disponibilizados pelos diversos servidores, através de seus protocolos. Deste modo, o sistema é bastante flexível porque a inserção de um novo protocolo implica na disponibilização de diversos serviços. Até mesmo a alteração de um protocolo já incluído no sistema pode trazer ao usuário novos serviços.

Uma das vantagens do sistema Exotica é que ele faz uma separação entre os dados de controle e os dados de aplicação. Os dados de controle são gerenciados pelas filas de mensagens persistentes, enquanto os dados de aplicação encontram-se replicados através de uma arquitetura *primary/backup*. Nesta arquitetura é utilizada também a noção de cluster de servidores, para que o impacto das falhas seja reduzido. Neste sistema, são empregados os mecanismos de *backup recovery* e *forward recovery*. Em RainMan apenas o mecanismo de *forward recovery* é empregado no nível de aplicação, para que no caso de ocorrência de falhas a execução de um workflow possa continuar.

A Tabela a seguir apresenta um resumo da comparação apresentada entre os sistemas de workflow descritos.

	Magi	Exotica	RainMan	Outro sistema [8]	CoAct	WorkToDo
Operação em modo desconectado	Não	Sim	Sim	Sim	Sim	Sim
Bloqueio	Não há bloqueio	Bloqueio de tarefas antes da desconexão	Tarefas são entregues aos executores à medida que vão sendo executadas	Bloqueio, que deve ser reatribuído a equipes periodicamente	Não há bloqueio	Bloqueio. Apenas determinadas tarefas podem ser executadas em modo desconectado
Meio de cooperação	Troca de mensagens	Fila de mensagens persistente	Listas de trabalho que são tratadas como objetos de rede endereçáveis	Mensagens enviadas pelos canais de eventos	Chamadas *import*, *save* e *delegate*	Através do Gerenciador de Processos e do Gerenciador de Instâncias
Coordenação	Realizada pelos servidores de serviços	Realizada através da fila de mensagens dos processos	Construtor representa ponto central de controle	Núcleo do sistema representa ponto de controle	Através do espaço de trabalho comum e dos espaços de trabalho privados de cada usuário	Por um ou mais Gerenciadores de Processos
Resolução de conflitos	Não ocorrem conflitos	Não ocorrem conflitos	-	Resolução pelo núcleo do sistema no momento em que recebe um evento de atualização	Responsabilidade do usuário no momento da troca de informação	Eventuais conflitos não são resolvidos

6 Abordagem Proposta

Nesta seção é apresentada a arquitetura proposta do sistema de gerência de workflow para redes com mobilidade e conectividade intermitente, e os conceitos envolvidos no funcionamento deste sistema.

6.1 Arquitetura do Sistema

A Figura 17 apresenta a arquitetura do sistema de workflow proposto neste trabalho. Ela é uma arquitetura intrinsicamente distribuída, projetada para o ambiente de redes móveis sem fio e onde podem estar presentes diversos usuários/equipes em colaboração.

De acordo com a figura apresentada, definimos basicamente três papéis com relação aos usuários executores de tarefas do workflow:

- usuários coordenadores na rede fixa (coordenadores secundários): são aqueles que possuem o poder de coordenação sobre as tarefas que são a eles delegadas. Um usuário coordenador pode delegar uma tarefa ou um grupo de tarefas para outros usuários/equipes, ficando também responsável por identificar e notificar a finalização da "tarefa maior", ou seja, do conjunto de tarefas sobre as quais tem responsabilidade;

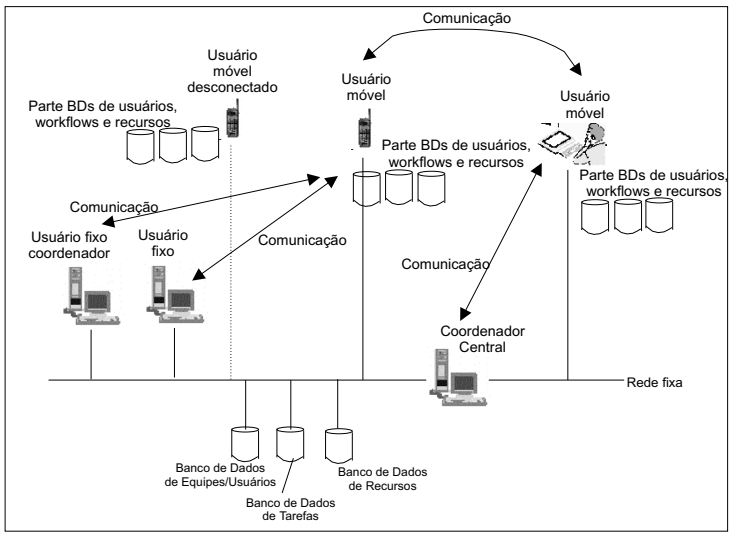

Figura 17: Arquitetura do sistema de gerência de workflow proposto.

- usuários na rede fixa sem poder de coordenação: são os usuários do sistema que apenas executam as tarefas a eles delegadas, não tendo nenhum direito de coordenação sobre elas. Por exemplo, estes usuários não podem delegar um subconjunto de suas tarefas para a execução por uma outra equipe;

- usuários na rede móvel: são os usuários de dispositivos móveis, cuja responsabilidade é executar as tarefas a eles delegadas. Estes usuários não possuem poder para delegar tarefas a outros usuários, porque quando desconectados eles ficam impossibilitados de controlar a cooperação de tarefas delegadas. Como assumimos conexão intermitente destes usuários com a rede, cada um deles possui em seu dispositivo local parte do banco de dados de tarefas e parte do banco de dados de recursos, para que possam operar em modo desconectado. Portanto, eles somente se conectam ao sistema quando necessitam receber ou transmitir alguma informação. Usuários na rede móvel podem se comunicar na rede fixa com o coordenador central e com os demais usuários desta rede. Também podem se comunicar com qualquer usuário na rede móvel.

O coordenador central é o módulo do sistema responsável pela gerência de todos os usuários, tarefas, recursos e da própria execução das diversas instâncias de workflow. Este módulo sempre mantém atualizadas nos bancos de dados as informações relativas à execução das instâncias. Além disso, ele possui o conhecimento de qual usuário está executando determinada tarefa, de qual usuário deve executar uma tarefa, de quais recursos estão sendo utilizados por quais usuários, de qual é o status da execução de uma instância e quais são as dependências entre todas as tarefas em execução. No momento da delegação de tarefas, o coordenador central pode optar por delegar uma mesma tarefa (ou conjunto de tarefas) a mais do que um usuário/equipe. A isso damos o nome de replicação de tarefas. Além disso, este coordenador pode optar por delegar poderes de coordenação a um usuário na rede fixa, dando a ele o direito de distribuir as tarefas que recebeu para a execução para outros usuários/equipes.

O coordenador central, além de manter o controle sobre toda a execução, é o responsável pela inicialização de qualquer instância de workflow. Depois de inicializada uma instância, ele deve delegar todas as tarefas aos usuários ou equipes que achar conveniente, de acordo com as informações contidas no banco de dados. No decorrer da execução, porém, pode ocorrer que estas sejam realocadas para outros usuários ou outras equipes a partir de um coordenador secundário na rede fixa. Esta realocação deve ser notificada ao coordenador central. Este é o único que tem acesso direto aos bancos de dados (de equipes/usuários, de workflows, de recursos) e que pode alterá-los. Todas as demais atualizações ou consultas devem ser feitas através deste módulo.

O banco de dados de equipes/usuários mantém o registro de todos os possíveis executores de tarefas. O banco de dados de tarefas armazena todas as tarefas dos workflows e as suas dependências. Dependências entre tarefas são descritas por meio de regras de dependência. Uma regra de dependência para um tarefa T informa de quais tarefas T depende e quanto tempo T deve esperar pela notificação da finalização das tarefas de que depende. Por fim, o banco de dados de recursos armazena a informação sobre todos os recursos disponíveis e necessários à execução de determinadas tarefas ou de determinados grupos de tarefas dos workflows.

6.2 Tipos de Tarefas

No sistema de gerência de workflow aqui proposto, classificamos as tarefas a serem executadas com relação a dois aspectos: replicação e desconexão. Se uma tarefa (ou um conjunto delas) foi replicada, isto significa que ela foi delegada para a execução por mais de um usuário.

Tarefas replicadas podem ser:

- excludentes: neste caso, apesar de terem sido delegadas a mais de um usuário/equipe para execução, a finalização da execução de uma instância desta tarefa implica no imediato cancelamento das demais instâncias. O controlador, que pode ser um coordenador secundário ou o coordenador central, apenas computa o resultado da execução desta primeira instância;

- conjuntas: neste caso, como existem várias instâncias da mesma tarefa em execução, a finalização de uma primeira instância já pode determinar o fim da execução da tarefa, mas não implica no cancelamento da execução das demais instâncias. Por exemplo, uma tarefa T_1 pode ser delegada para os usuários U_2, U_3 e U_5. Se U_5 é o primeiro a finalizar a tarefa e notificar o coordenador central (ou secundário) sobre a sua finalização, a tarefa T_1 é marcada como executada, mas o coordenador pode ser posteriormente informado a respeito do término de execução das demais instâncias desta tarefa. A notificação de término pode incluir, além do estado final da execução da tarefa, também os recursos produzidos ou modificados durante a sua execução.

Assim como no caso de tarefas excludentes mensagens devem ser enviadas aos executores para cancelamento da execução quando a primeira instância terminar, no caso de tarefas conjuntas mensagens devem ser enviadas de todos os executores aos seus coordenadores para notificação do término da execução.

Com relação à desconexão, tarefas podem ser:

- passíveis de serem executadas em modo desconectado (tarefas "desconectáveis"): são aquelas tarefas definidas como podendo ser executadas por um usuário durante uma desconexão, assim como ocorre em WorkToDo (veja seção 5). Neste caso, além da definição da tarefa e das dependências dela com as demais, também os recursos associados devem ser transferidos para o cache do usuário antes da desconexão. Caso contrário, estas tarefas não poderão ser executadas até que o executor volte a se conectar ao sistema;
- não passíveis de serem executadas em modo desconectado (tarefas "não-desconectáveis"): tarefas assim definidas possivelmente possuem restrições rígidas de tempo, de modo que o coordenador (central ou secundário) não pode esperar um tempo indeterminado para obter o resultado da computação, ou são tarefas que possuem uma quantidade grande de recursos associados, os quais não poderiam ser carregados em um dispositivo móvel. Além disso, podem ser tarefas das quais muitas outras tarefas dependam. Se estas fossem definidas como tarefas "desconectáveis", talvez grande parte dos usuários ficaria em espera um longo período de tempo.

O quadro seguinte apresenta como uma tarefa, por exemplo T_4, pode ser descrita no sistema, incluindo seus atributos e suas dependências.

```
Tarefa T4
 // atributos estáticos
   Título: Convocação de bombeiros
   Descrição: Convocar o corpo de bombeiros...
   Dependências:
        Depende de: T5
   Atributos:
        Máxima Duração: 24 horas
        Replicação: (sim, excludente)
        Mobilidade: sim
   Recursos: R1, R2
   Valor default: não executada
   Saída: Documento X
 // atributos dinâmicos
   Usuários dos quais deve receber
            notificação: U4
   Tempo de espera de notificação:
   Status: iniciado/ concluído/ nil
   Executor: U1
```

A descrição apresentada é informal e se destina apenas a tornar claro quais são os atributos de uma tarefa e quais deles são conhecidos em tempo de execução. Essa tarefa T_4 possui um nome e uma descrição, depende de T_5, deve ser executada em um prazo máximo de 24 horas, pode ser replicada e é excludente, pode ser executada em modo desconectado e produz como saída um documento X. Em tempo de execução, são conhecidos para esta tarefa o seu estado e o usuário executor, no caso U_1.

6.3 Execução de Workflows

Quando inicia o processo de execução de uma instância de um workflow, o coordenador central escolhe alguns usuários ou algumas equipes para a execução de determinadas tarefas. Se os usuários/equipes são também coordenadores, eles podem delegar estas tarefas a outros membros do grupo.

Usuários na rede fixa devem notificar o módulo coordenador a respeito do resultado da computação e dos recursos produzidos ou modificados a cada vez que uma tarefa é executada com sucesso. No caso dos usuários da rede móvel, como a conexão é intermitente e estes a princípio apenas se conectam para enviar ou receber mensagens ou também dados/recursos, o módulo coordenador não recebe imediatamente a notificação do resultado da computação. Dessa forma, em determinados momentos este módulo não conhece o real estado de execução de parte do workflow. Assim que se reconectam à rede, os usuários móveis devem enviar suas notificações a respeito do resultado da execução das tarefas.

Por outro lado, se o usuário móvel U_1 está subordinado diretamente a um usuário U_2 na rede fixa (U_2 é coordenador secundário e delegou tarefas a U_1) e não diretamente ao coordenador central, este usuário U_1 deve notificar o usuário U_2 e este sim por sua vez repassar a notificação ao coordenador central. Nos casos em que usuários móveis estão subordinados a usuários na rede fixa, se um usuário U_3 (fixo ou móvel) precisa conhecer o estado da computação de uma tarefa em execução por U_1 e não recebeu ainda a notificação deste, então U_3 pode consultar o usuário U_2. No entanto, eventualmente U_3 não tem conhecimento do usuário ao qual U_1 está subordinado (U_2). Desta forma, U_3 deve primeiro consultar o coordenador central, que então o informa que U_1 está subordinado a U_2. Se nem mesmo U_2 recebeu a notificação de U_1, U_2 deve autorizar um plano alternativo de execução, para que U_3 possa continuar sua computação. Este plano possivelmente estará baseado em valores default para as tarefas. Desse modo, U_3 seria informado por U_2 de que o resultado da computação não é conhecido e que, portanto, deve assumir o valor default para um recurso que deveria ter sido o resultado de U_1. Isto define um controle descentralizado baseado no conceito de delegação de tarefas que reduz a carga sobre o coordenador central.

```
U2 delegou tarefa para U1
U1 deve notificar U2 sobre seu estado de computação
U3 deve executar uma tarefa que depende da execução de outra tarefa
   por U1
Se U3, por algum motivo, não recebeu comunicação de U1, U3 deve consultar
   coordenador central para saber qual usuário coordena U1
Se U2 também não recebeu notificação de U1, então U2 deve adotar
   uma política de modo que U3 possa continuar sua computação
```

Quando usuários estão executando suas tarefas na rede fixa, eles podem consultar os bancos de dados de informações através do coordenador central. No entanto, no caso da rede móvel, como assumimos conexão intermitente, cada usuário deve ter conhecimento das informações relativas às tarefas que precisa executar, por exemplo seus recursos e as tarefas das quais as suas tarefas dependem. Além disso, usuários móveis devem conhecer quais usuários estão associados às tarefas dependentes das suas tarefas.

Por exemplo, considere um usuário U_1 na rede móvel. Este usuário deve executar as

tarefas T_2, T_3 e T_4, as quais são todas tarefas "desconectáveis". Considere que T_2 e T_3 não dependem de nenhuma outra tarefa, mas que T_4 depende do resultado da computação de uma tarefa T_5, que está sendo executada por um usuário U_4 na rede fixa ou mesmo na rede móvel. Dessa forma, U_1 sabe que deve receber de U_4 uma notificação a respeito da execução de T_5.

A Figura 18 representa o problema de comunicação abordado.

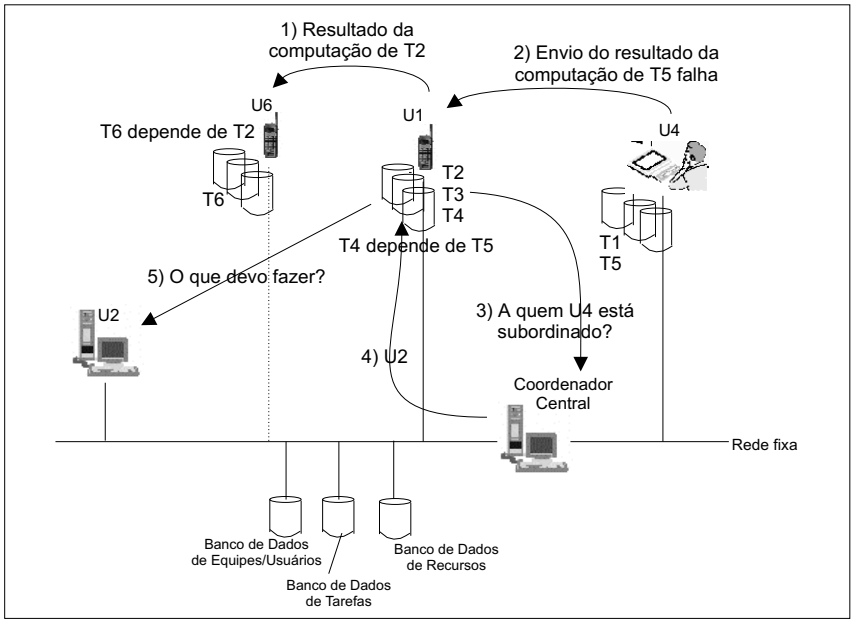

Figura 18: Exemplo de comunicação entre usuários no sistema de gerência de workflow proposto.

Porém, se durante o período de espera determinado na especificação de dependências da tarefa T_4 o usuário U_1 não tiver recebido a notificação do resultado da computação de T_5, provavelmente o usuário U_4 não conseguiu completar T_5 a tempo ou não conseguiu enviar sua notificação. Se o problema foi o primeiro, então U_1 deve consultar o coordenador para obter informações de como proceder, obtendo, por exemplo, a resposta de que deve continuar esperando.

Se a falha tiver sido no envio da notificação por parte de U_4 para U_1, então U_4 precisa notificar o coordenador a respeito do ocorrido e do estado da sua computação, para que quando consultado este possa fornecer a informação correta.

Considere agora que da tarefa T_2 do usuário U_1 dependa a execução da tarefa T_6 do usuário U_6. Neste caso, como U_1 detém esta informação em seu próprio dispositivo, a notificação é enviada diretamente para U_6 ao término de T_2 e também para o coordenador central. Isto garante que o coordenador central tem a informação mais recente sobre o estado da computação do workflow, exceto nos momentos em que ainda os coordenadores secundários não enviaram as notificações relativas à execução das tarefas pelos usuários que coordenam.

6.4 Resolução de Conflitos

Tanto usuários da rede móvel quanto usuários da rede fixa devem armazenar em cache local o registro das atualizações realizadas localmente e dos resultados das tarefas executadas. O registro das tarefas executadas deve ser retirado do cache apenas depois que o usuário coordenador notifica o usuário executor de que o registro tornou-se persistente no módulo coordenador. Se recursos foram criados nos caches locais, eles devem ser transferidos para os coordenadores secundários, os quais repassam para o módulo coordenador (central ou secundário).

No entanto, se os registros de atualização de dois ou mais usuários, quando chegarem ao coordenador secundário (quando este for o caso), forem conflitantes (decorrentes da política otimista de concorrência), este deve ser capaz de resolver o conflito e enviar o registro de atualização conforme a política adotada, de modo que o coordenador central apenas execute este registro no banco de dados na rede fixa, sem se preocupar com a ocorrência de possíveis conflitos.

6.5 Exemplo de Aplicação

Considere que o sistema proposto seja empregado no âmbito de um sistema de controle e notificação de uma emergência, como por exemplo um incêndio em uma determinada região. O workflow que descreve as atividades a serem executadas para o combate ao incêndio é composto das 5 tarefas apresentadas na Figura 18, que mostra a distribuição e as dependências entre as tarefas. Segue apenas uma descrição de quais seriam estas tarefas.

```
T1: Verificar as proporções do incêndio
T5: Registrar as proporções do incêndio -> saída: Documento Y
T2: Avisar o coordenador do meio ambiente da região sobre o incêndio
T3: Verificar a quantidade de homens do corpo de bombeiros
    disponíveis
T4: Convocar o corpo de bombeiros, repassando a quantidade de
    homens/equipamentos necessários para combater incêndio
T6: Compor nota à imprensa conforme requerido pela coordenação de
    meio ambiente
```

De acordo com o workflow descrito, se o usuário U_1 não receber de U_4 a informação do tamanho da área afetada (informação esta apresentada no documento Y), o coordenador central deve apresentar uma solução a U_1 para a execução de T_4, por exemplo indicando/definindo para T_4 o valor default como a capacidade máxima de homens e equipamentos do corpo de bombeiros para o combate ao incêndio.

Nesse exemplo apresentado, as tarefas T_2, T_3 e T_4 podem ser tarefas replicáveis e excludentes. No caso de T_2, quando o coordenador do meio ambiente fosse avisado, deveria ser cancelada a execução desta tarefa pelos demais usuários.

No entanto, T_6 poderia ser uma tarefa conjunta. Neste caso, quando o coordenador recebesse todas as notas compostas pelos executores, se elas fossem conflitantes, ele poderia optar, por exemplo, por: (1) aceitar o que está válido conforme a maioria dos executores descreveu; (2) dirigir um quórum para que todos os executores chegassem a um acordo.

6.6 Comparação entre Abordagens

Esta seção tem como objetivo comparar de forma sucinta a abordagem aqui proposta com as demais abordagens apresentadas na seção 5.

Existem basicamente duas diferenças principais entre a nossa abordagem e a abordagem da maioria dos sistemas de workflow aqui apresentados. São elas: a questão do bloqueio e a questão da delegação de tarefas.

Em Exotica e WorkToDo, por exemplo, a cada usuário está associada uma lista de trabalho com as tarefas que eles podem executar. Se uma tarefa é selecionada ou bloqueada para execução em modo desconectado por um usuário, então esta tarefa é retirada da lista dos demais.

Na abordagem aqui proposta, uma tarefa pode ser replicada para vários usuários, de modo que ela possa ser executada por mais de um deles. Neste caso, define-se como o módulo coordenador deve proceder ao receber os resultados das tarefas replicadas.

Além disso, como tarefas são explicitamente delegadas aos usuários/equipes, não há como usuários escolherem as tarefas que desejam executar, como ocorre em WorkToDo, por exemplo. Em nossa abordagem esta decisão de alocação de usuários/equipes para tarefas é de responsabilidade do(s) coordenador(es).

Com relação ao bloqueio, a nossa abordagem assume uma política otimista. Isso porque, em um sistema cooperativo em um ambiente de rede com desconexões freqüentes o bloqueio de tarefas que sejam casualmente precedentes a outras pode gerar um atraso ou até mesmo a interrupção de todo o workflow.

Assim como na maioria dos trabalhos, a nossa arquitetura também permite a execução no modo desconectado. Nela, a cooperação ocorre através da troca de mensagens entre os usuários. Cada mensagem pode conter, além do resultado da computação (estado final da execução da tarefa), também informações a respeito dos recursos criados ou modificados.

A coordenação na arquitetura proposta é dividida entre o coordenador central e os coordenadores secundários. O coordenador central possui a responsabilidade sobre o controle da execução dos workflows. Ele tem o conhecimento de quais tarefas estão sendo executadas por cada usuário. Os coordenadores secundários, no entanto, são os responsáveis pela resolução de conflitos. Quando uma notificação de modificação de um recurso chega ao coordenador central, ele apenas torna esta modificação persistente no banco de dados, assumindo que os possíveis conflitos já foram resolvidos.

7 Conclusões

Sistemas de workflow representam uma classe de sistemas de suporte a colaboração bastante importante no âmbito empresarial. Estes sistemas possuem natureza distribuída e são claramente aplicáveis ao ambiente de computação móvel.

Uma vez tendo definida a arquitetura do sistema e o seu modelo de computação, o próximo passo será o projeto, incluindo protocolos de comunicação, sincronização e resolução de conflitos, e a implementação do sistema. O objetivo será o de validar o modelo proposto em cenários de workflow para redes móveis e medir como o grau de replicação de tarefas pode agilizar a execução de workflows em situações que requerem ações imediatas.

Referências

[1] G. Alonso, D. Agrawal, Amr El Abbadi, C. Mohan, R. Gunthor, and M. Kamath. Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management. In *Proceedings of the IFIP WG8.1 Working Conference on Information Systems Development for Decentralized Organizations*, Trondheim, Norway, August 1995.

[2] G. Alonso, B. Reinwald, and C. Mohan. Distributed Data Management in Workflow Environments. In *In Seventh International Workshop on Research Issues in Data Engineering (RIDE'97)*, Birmingham, England, April 1997.

[3] B. Bhushan and A. Patel. Requirements and the Concept of Cooperative System Management. *International Journal of Network Management*, 8:139-158, 1998.

[4] G. A. Bolcer. Magi: An Architecture for Mobile and Disconnected Workflow. In *IEEE Internet Computing*, pages 46-54, May-June 2000.

[5] P. H. Carstensen. *Computer Supported Coordination.* PhD thesis, Riso National Laboratory, Roskilde, Denmark, April 1996.

[6] T. A. S. Coelho and M. Endler. Gerenciamento de Workflow em Redes com Mobilidade e Conectividade Intermitente. Artigo aceito no IV Workshop de Comunicação Sem Fio (WCSF2002), a ser realizado de 23-25 de Outubro deste ano, em São Paulo.

[7] A. J. Dix. Cooperation Without (Reliable) Communication: Interfaces for Mobile Applications. *Distributed Systems Engineering*, 2(3):171-181, 1995.

[8] H. J. Domingos, J. L. Martins, N. M. Preguiça, and S. M. Duarte. A Workflow Architecture to Manage Mobile Collaborative Work. Primeiro Encontro Português de Computação Móvel (EPCM'99), November 1999.

[9] D. Hollingsworth. The Workflow Management Coalition Specification - The Workflow Reference Model. Technical Report TC00-1003, Workflow Management Coalition, Hampshire, UK, January 1995.

[10] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *Thirteenth ACM Symposium on Operating Systems Principles*, 25(5):213-225, 1991.

[11] J. Klingemann, T. Tesch, and J. Wasch. Enabling Cooperation among Disconnected Mobile Users. In *Second IFCIS International Conference on Cooperative Information Systems (CoopIS'97)*, pages 36-46, Kiawah Island, South Carolina, USA, June 1997.

[12] C. Mohan, G. Alonso, R. Gunthor, and M. Kamath. Exotica: A Research Perspective on Workflow Management Systems. *Data Engineering Bulletin*, 18(1):19-26, 1995.

[13] S. Paul, E. Park, and J. Chaar. RainMan: A Worflow System for the Internet. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 159-170, Monterey, California, December 1997.

[14] E. Pitoura and G. Samaras. *Data Management for Mobile Computing.* Kluwer Academic Publishers, 1998.

[15] F. Ranno and S. K. Shrivastava. A Review of Distributed Workflow Management Systems.

[16] L. H. Reinehr and M. B. F. de Toledo. WorkToDo - Um Sistema de Gerenciamento de Workflows para Ambientes de Comunicação sem Fio. In *20º Simpósio Brasileiro de Redes de Computadores*, pages 619-634, Búzios, Rio de Janeiro, 2002.

[17] T. Rodden. A Survey of CSCW Systems. *Interacting with Computers*, 3(3):319-353, 1991.

[18] T. Rodden and G. Blair. CSCW and Distributed Systems: The Problem of Control. In *ECSCW'91*, Amsterdam, September 1991.

[19] M. M. Theimer, A. J. Demers, K. Petersen, M. J. Spreitzer, D. B. Terry, and B. B. Welch. Dealing with Tentative Data Values in Disconnected Work Groups. In *Proceedings of the First IEEE Workshop on Mobile Computing Systems and Applications*, pages 192-195, Santa Cruz, California, December 1994.