WorkToDo – Um Sistema de Gerenciamento de Workflows para Ambientes de Comunicação sem Fio

Leonardo Hartleben Reinehr, **Maria Beatriz Felgar de Toledo**
Instituto de Computação - Universidade Estadual de Campinas
Caixa Postal 6176, 13083-970, Campinas - SP - Brasil
e-mail: {leonardo.reinehr, beatriz}@ic.unicamp.br

Resumo

Este artigo aborda o problema da utilização de Sistemas de Gerenciamento de Workflows (SGWFs) em ambientes de comunicação sem fio, onde as conexões são instáveis e não permanentes. Nesse ambiente um usuário poderia executar tarefas independentemente de sua localização e tipo de conexão, preservando sua autonomia. O modelo proposto visa atender esses requisitos através da reserva de tarefas e transferência antecipada de dados/aplicações para o computador móvel.

Abstract

This paper addresses the problem of using Workflow Management Systems (WFMS) in wireless communication environments, where connections are unstable and non-permanent. In such an environment, users should be able to perform tasks independently from location and type of connection, in order to have their autonomy preserved. The proposed model aims at meeting these requirements through task reservation and antecipated transfer of data/applications to the mobile computer.

Palavras-chave: Sistemas de Gerenciamento de Workflows, Computação Móvel, Comunicação sem Fio.

1 Introdução

Sistemas de Gerenciamento de *Workflows* (SGWFs) têm provocado grande interesse devido à sua capacidade de aumentar a eficiência de uma organização através da automatização de seus processos. Diversos SGWFs foram implementados comercialmente e passaram a ser utilizados com sucesso. Entretanto ainda apresentam limitações, como por exemplo baixa interoperabilidade entre diferentes SGWFs, suporte insuficiente para recuperação de falhas e pouca flexibilidade de uso [1, 4, 5, 8, 9, 11].

Normalmente os SGWFs exigem que os usuários possuam uma conexão permanente por meio de uma rede local (como por exemplo Ethernet). Esta é uma restrição severa, pois obriga os usuários a utilizarem o sistema somente em um determinado local. Considerando a grande disseminação de redes sem fio, é interessante permitir que os usuários utilizem o SGWF sem

restrições de localização e sem exigências de conexões permanentes, confiáveis e de alta velocidade. Isso expande sensivelmente a flexibilidade de uso de um SGWF, pois os usuários podem executar as tarefas que lhes cabem a partir de seus *laptops* ou de seus computadores pessoais, independentemente de sua localização e do tipo de sua conexão ao SGWF.

Consideremos o seguinte exemplo. Uma empresa que presta serviços de manutenção para diversos clientes espalhados em uma região possui um processo gerenciado por um SGWF, composto por diversas tarefas, entre elas `VisitarCliente`. Essa tarefa é executada por um técnico de manutenção, que a partir de uma ordem de serviço se desloca até o cliente e executa o serviço de manutenção. Em um SGWF convencional, no início do dia o técnico reúne as ordens de serviço atribuídas a ele, visita os clientes e ao final do dia retorna à empresa e informa ao SGWF os resultados de cada visita. Suponhamos que, depois de o técnico deixar a empresa, cheguem novas ordens de serviço. O técnico somente tomará conhecimento delas no dia seguinte e, dependendo da localização dos clientes, perderá um tempo significativo.

Por outro lado, se o SGWF oferecesse suporte à mobilidade, com um *laptop* o técnico poderia se conectar ao SGWF e verificar as ordens de serviço atribuídas a ele. Ao terminar um serviço o técnico se conectaria novamente ao SGWF para relatar o resultado da visita, aproveitando para verificar a existência de novas ordens de serviço. O técnico receberia os serviços de novos clientes e poderia alterar seu roteiro de visitas, economizando tempo.

Esse exemplo mostra como a operação através de redes sem fio expande a flexibilidade de uso de um SGWF, além de aumentar sua eficiência. Entretanto, com esse tipo de operação o gerenciamento de processos torna-se mais complexo. Quando o usuário está sempre conectado ao SGWF, pode ser constantemente monitorado e pode receber notificações de novas tarefas a qualquer momento. Já no ambiente de redes sem fio isso não é possível, pois o usuário pode permanecer longos períodos desconectado do sistema. Além disso, falhas de comunicação tornam-se muito mais freqüentes. Portanto, novos mecanismos de controle se fazem necessários, para que o SGWF respeite a autonomia dos usuários mas ao mesmo tempo garanta que os processos sejam executados de acordo com sua especificação.

Neste artigo apresentamos o WorkToDo, um SGWF para ambientes de comunicação sem fio. O WorkToDo permite que os usuários executem tarefas através de conexões instáveis e de baixa largura de banda. Mais do que isso, os usuários podem, tomando certas medidas, executar tarefas sem estarem conectados ao SGWF.

O artigo está organizado da seguinte forma. A Seção 2 relaciona este trabalho com outros propostos. A Seção 3 descreve o sistema WorkToDo. A Seção 4 contém aspectos do protótipo implementado, juntamente com um exemplo. Finalmente, a Seção 5 apresenta as conclusões.

2 Trabalhos Relacionados

Existem trabalhos de pesquisa que abordam a questão da utilização de SGWFs em ambientes de computação móvel. O Exotica [1] modifica o modelo e arquitetura originais para permitir a operação desconectada. Antes de se desconectar o usuário seleciona as tarefas que deseja executar e os dados relativos a elas são copiados para sua máquina. Isso é feito através da operação de *lock*, pela qual um usuário reserva tarefas, retirando-as das listas de trabalho dos demais usuários. Durante a operação desconectada o usuário executa as tarefas selecionadas. No momento da reconexão os dados são atualizados junto ao SGWF.

O modelo INCAS [2] não exige que as entidades que executam as tarefas possuam uma conexão permanente ao SGWF ou que as conexões efetuadas sejam de alta velocidade e estáveis, caracterizando um ambiente de comunicação sem fio. O INCAS exige apenas que as entidades sejam capazes de receber um INCA, executar as operações requisitadas por ele e redirecioná-lo para a próxima entidade. Mas há alguns problemas de implementação relacionados exatamente a essa última condição (por exemplo: se uma entidade X deve redirecionar um INCA para outra entidade Y, o que X deve fazer se Y não puder ser contactada por muito tempo?), para os quais ainda não foram apresentadas soluções.

Büssler [3] relaciona uma série de requisitos para a incorporação da operação desconectada em SGWFs convencionais, apresentando o conceito de *mobile worklists* para tratar tais requisitos. Não há, entretanto, referência a alguma implementação que comprove a viabilidade e eficiência desse mecanismo. E Jing [8] discute a técnica do *prefetching* em ambientes móveis.

Nossa abordagem visa apresentar não apenas um modelo e uma arquitetura para ambientes de comunicação sem fio, mas também um protótipo capaz de validá-los e verificá-los. O SGWF desenvolvido aborda os aspectos relativos a esse tipo de ambiente, provendo suporte à operação desconectada. Como o Exotica, utilizamos a abordagem do usuário selecionar as tarefas que deseja executar em modo desconectado através de *locks*. Além disso incorporamos alguns dos requisitos citados por Büssler, como gerenciamento local de prazos de tarefas e cópia dos dados a elas relacionados. Diferentemente desses trabalhos, que enfocam somente a operação desconectada em SGWFs, o modelo aqui proposto também trata os problemas existentes em ambientes sem fio, com desconexões freqüentes e baixa largura de banda.

3 WorkToDo

O ***WorkToDo*** é um Sistema de Gerenciamento de Workflows que oferece suporte para ambientes de comunicação sem fio, permitindo que um usuário execute tarefas sem estar conectado ao SGWF (operação desconectada) e conectado através de uma conexão instável e de baixa largura de banda (operação semi-conectada). O modelo e a arquitetura do WorkToDo são descritos nas seções seguintes.

3.1 Modelo de Processo

A modelagem de um processo consiste em definir os seguintes aspectos: que tarefas devem ser executadas, em qual momento, por quem e com quais dados. Esses aspectos são chamados, respectivamente, de funcional, comportamental, organizacional e informacional [4, 7].

A Figura 1 mostra uma representação do modelo utilizado pelo WorkToDo. Um processo é formado por um conjunto de atividades e dependências entre elas, e cada atividade possui um conjunto de dados de entrada e saída e uma entidade processadora responsável por sua execução.

O WorkToDo utiliza o conceito de ***Tipos de Processo***. Isso significa que quando um processo é modelado, o que se define é na verdade um tipo de processo, que pode então ser instanciado várias vezes, criando execuções particulares desse tipo.

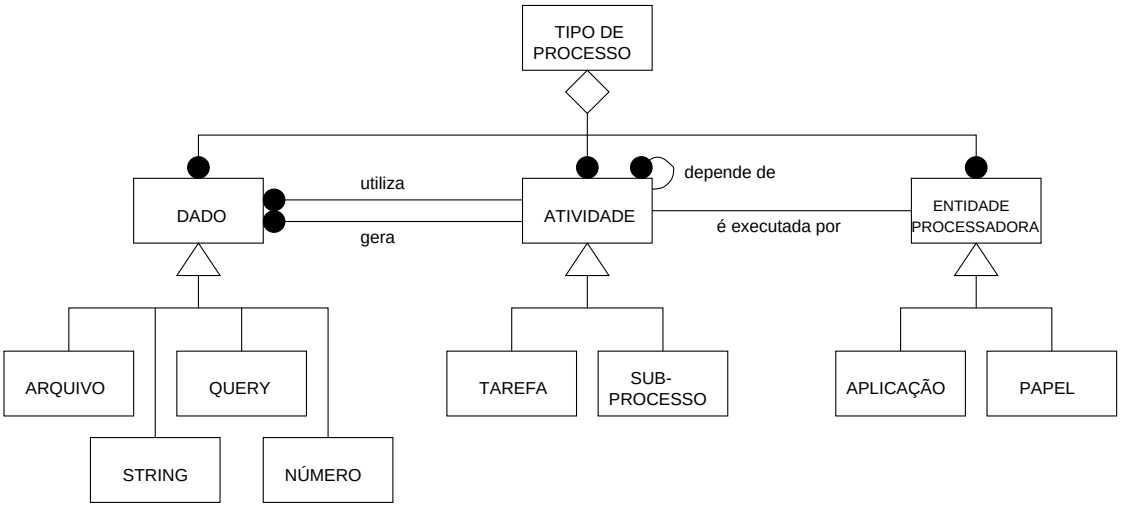

Figura 1: Modelo WorkToDo.

3.1.1 Atividades

Um processo é formado por um conjunto de atividades, de tal forma que quando tais atividades são desempenhadas a execução do processo está concluída, definindo portanto o aspecto funcional de um processo. As atividades podem ser tarefas ou sub-processos. As tarefas são as ações elementares de um processo, enquanto os sub-processos são outros processos.

3.1.2 Tarefas

As tarefas são aquelas atividades que não podem mais ser decompostas, representando ações elementares. São definidas através dos ***Modelos de Tarefas***, os quais são instanciados pelos processos. Os dados relativos às tarefas, por sua vez, são especificados na definição do processo, de forma que cada instância de um modelo de tarefa utiliza dados próprios.

As tarefas podem diferir umas das outras, exigindo tratamentos distintos por parte do sistema. Por isso elas são classificadas em três tipos:

Automáticas. São executadas por algum software invocado automaticamente pelo SGWF (por exemplo: acessar um banco de dados, copiar um arquivo).

Semi-automáticas. São executadas por um humano auxiliado por um sistema de software (por exemplo: redigir um documento em um editor de textos, escanear uma foto).

Manuais. São executadas exclusivamente por um humano (por exemplo: preencher um formulário, enviar uma carta pelo correio).

São definidos cinco estados possíveis para uma tarefa: `NOT_READY` – a tarefa ainda não pode ser executada porque as condições necessárias para sua execução não foram satisfeitas; `READY` – as condições necessárias para a execução da tarefa foram satisfeitas e portanto ela está pronta para ser executada; `RUNNING` – a tarefa está em execução; `SUCCEEDED` – a tarefa foi executada com sucesso; `FAILED` – ocorreu alguma falha durante a execução da tarefa.

A transição entre esses estados é mostrada na Figura 2. `NOT_READY` é o estado inicial, enquanto `SUCCEEDED` e `FAILED` são os estados finais. Em um dado momento, uma tarefa pode se encontrar em um e somente um estado.

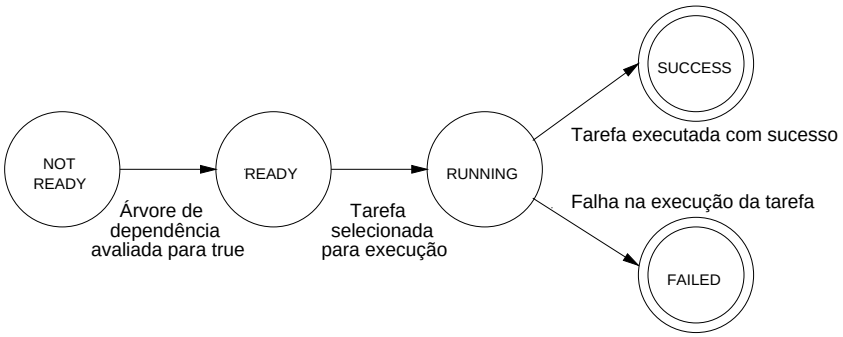

Figura 2: Diagrama de transição dos estados das tarefas.

3.1.3 Regras de Dependência

Para modelar o aspecto comportamental de um processo, o WorkToDo utiliza regras de dependência. Uma regra de dependência contém as condições que devem ser satisfeitas para que determinada tarefa seja executada. Estas condições são descritas em termos de transições de tarefas para estados. Uma regra é composta por zero ou mais termos, onde cada termo é formado por um nome de tarefa e um estado. Os termos podem se relacionar através dos operadores booleanos `and` e `or`. Um exemplo de regra de dependência é mostrado abaixo:

$$\texttt{and } (t_1 \rightarrow \texttt{ SUCCEEDED, } t_2 \rightarrow \texttt{ SUCCEEDED})$$

Nesse exemplo, a tarefa associada à regra somente será executada se as tarefas t_1 e t_2 alcançarem o estado `SUCCEEDED`. A regra de dependência possui dois termos: $t_1 \rightarrow$ `SUCCEEDED` e $t_2 \rightarrow$ `SUCCEEDED`, relacionados através da operação `and`. Os estados válidos para as regras são aqueles possíveis para as tarefas, conforme a Seção anterior. Uma regra de dependência vazia significa que a tarefa não depende de nenhuma condição para ser executada.

3.1.4 Entidades Processadoras

Para toda tarefa existe uma ***Entidade Processadora*** responsável por sua execução. As entidades processadoras modelam o aspecto organizacional de um processo e são definidas nos modelos de tarefas. O WorkToDo define dois tipos de entidades processadoras:

Aplicações. São as entidades processadoras das tarefas automáticas. Para executar a tarefa a aplicação correspondente deve ser invocada (possivelmente com parâmetros e/ou dados de entrada/saída). A cada tarefa automática é atribuída uma aplicação.

Usuários. São as entidades processadoras de tarefas semi-automáticas ou manuais. A atribuição de tarefas aos usuários se dá através de um sistema de papéis. Um ***Papel*** designa um grupo de usuários que possuem determinadas características. Na especificação da tarefa é definido um papel R, de maneira que qualquer usuário pertencente a R pode executar a tarefa.

3.1.5 Dados

As tarefas podem utilizar dados de entrada e gerar dados de saída, dados estes que correspondem ao aspecto informacional dos processos. No WorkToDo, tais dados podem ser de quatro

tipos: Arquivos – arquivos armazenados em disco; *Queries* – expressões em SQL (*Simple Query Language*) que definem consultas a bancos de dados e que possivelmente retornam um conjunto de dados (registros); Strings – valores alfanuméricos; e Números – valores inteiros ou reais.

Cada processo em execução utiliza uma área exclusiva, denominada ***Contexto do Processo***, na qual armazena seus dados. Assim, quando uma tarefa necessita recuperar ou atualizar dados, o faz nessa área. O contexto de uma instância de processo P somente é acessível às tarefas pertencentes a P, isolando assim os dados de cada instância de processo e evitando que a execução de um processo interfira em outro.

3.1.6 Linguagem de Definição

O WorkToDo utiliza uma ***Linguagem de Definição de Processo – LDP***, através da qual é possível definir tipos de processos, modelos de tarefas e aplicações. A definição de um tipo de processo contém uma lista de atividades e, para cada atividade, seus dados de entrada e saída, suas dependências e sua entidade processadora. Um modelo de tarefa especifica as características da tarefa, como tipo e prioridade. Já a definição de uma aplicação contém as características da aplicação, como tamanho e sistema operacional no qual pode ser executada.

3.2 Arquitetura do Sistema

O WorkToDo possui uma arquitetura distribuída, mostrada na Figura 3, cujos componentes estão descritos nas seções seguintes. Como cada um desses componentes pode ser executado em uma máquina diferente, a comunicação entre eles é efetuada utilizando alguma plataforma que forneça um método de comunicação remota, como RPC (*Remote Procedure Call*), Sockets, CORBA (*Common Object Request Broker Architecture*) ou RMI (*Remote Method Invocation*).

3.2.1 Gerenciador de Usuários e Papéis

O Gerenciador de Usuários e Papéis (*User and Role Manager — URM*) é o componente responsável por gerenciar usuários e papéis dentro do sistema. Verificações como os papéis aos quais um determinado usuário pertence e a lista de usuários de um determinado papel são efetuadas pelo URM, além da adição e remoção de usuários e papéis. Para tanto, este componente controla dois repositórios:

Usuários. Contém informações a respeito dos usuários do sistema: nome, senha, situação (conectado ou não ao SGWF), endereço IP de sua máquina, há quanto tempo está conectado e quando foi sua última conexão.

Papéis. Armazena os papéis existentes, juntamente com uma lista dos usuários pertencentes a cada papel. Um usuário pode pertencer a zero ou mais papéis.

3.2.2 Gerenciador de Definições

A função do Gerenciador de Definições (*Definition Manager — DM*) é gerenciar as definições de processos, tarefas e aplicações. O DM permite recuperar as definições existentes, além de adicionar e remover definições. Para tanto, mantém três repositórios que armazenam os tipos de processos, os modelos de tarefas e as aplicações.

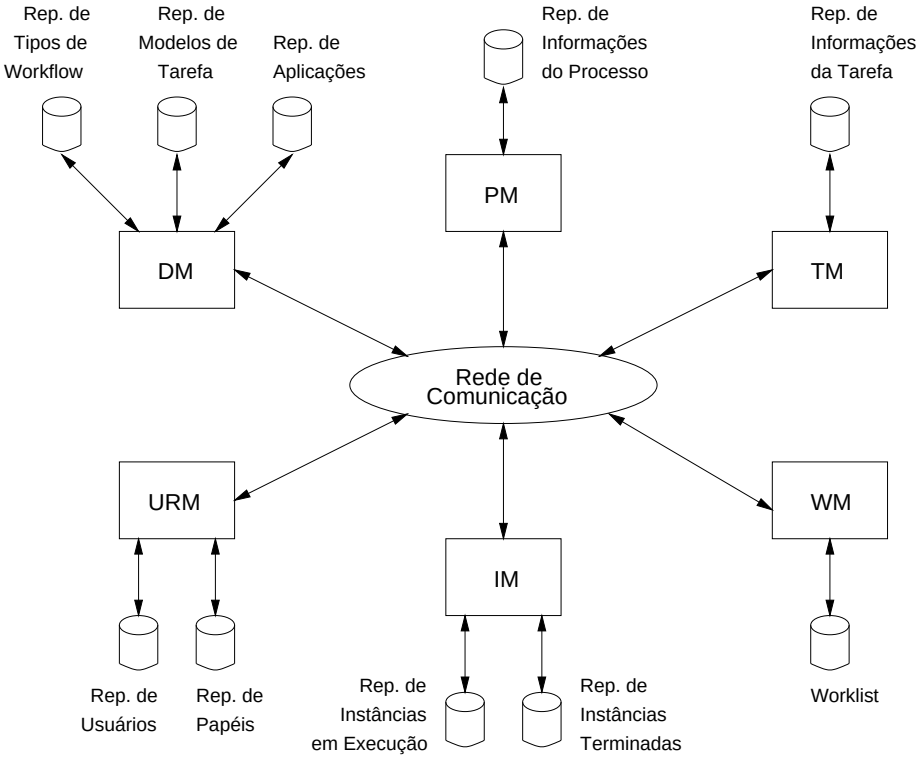

Figura 3: Arquitetura WorkToDo.

Quando um tipo de processo é adicionado ao repositório de tipos, o DM requisita o nome de um papel. Esse papel é chamado ***Papel Criador*** do tipo de processo. Instâncias desse tipo somente poderão ser criadas por usuários que pertencem ao papel criador.

3.2.3 Gerenciador de Instâncias

O Gerenciador de Instâncias (*Instance Manager — IM*) tem a função de manter um registro das instâncias de processo em execução, bem como daquelas cuja execução já foi concluída. O IM também é responsável por criar um Gerenciador de Processo para cada instância criada; no momento da criação, o IM transfere para o Gerenciador de Processo uma cópia da definição do processo. O IM mantém dois repositórios:

Processos. Armazena informações a respeito das instâncias em execução, como nome, tipo e máquina a partir do qual a instância está sendo gerenciada.

Processos completados. Armazena informações a respeito das instâncias cujas execuções já foram concluídas. Além do nome da instância, guarda seu estado final, contendo as datas e horas de início e término da execução e o estado final de cada uma de suas tarefas.

O IM também é responsável por tratar eventuais falhas dos Gerenciadores de Processo. Em tais casos, o IM pode criar um novo PM em outra máquina, inicializando-o com o último estado consistente do PM que falhou. Por isso, o IM armazena no repositório de Processos o estado atual de cada instância em execução, estado esse atualizado periodicamente.

3.2.4 Gerenciador de Processo

O Gerenciador de Processo (*Process Manager — PM*) é o componente responsável por coordenar a execução de um processo. O PM verifica quais tarefas estão prontas para executar, inicia sua execução e coleta seus resultados, verificando se tais resultados permitem a execução de novas tarefas. Existe um PM para cada instância de processo em execução.

O PM mantém uma lista com as tarefas que compõem o processo, denominada ***Tasklist***. A *tasklist* armazena as informações necessárias para a execução das tarefas, como nome, tipo, entidade processadora, estado atual e dados utilizados como entrada e saída. Esse conjunto de informações recebe o nome de ***Definição de Tarefa*** (*Task Definition*). A *tasklist* constitui-se, portanto, de uma lista de *task definitions*. O PM mantém ainda um repositório que armazena o estado da instância do processo, contendo informações como o nome do processo e seu horário de início e de término, além da *tasklist*.

O PM contém dois sub-componentes que auxiliam no gerenciamento da instância:

Escalonador (*Scheduler*). Avalia as condições para a execução das tarefas, de acordo com suas respectivas regras de dependência. É o Escalonador, portanto, quem decide quais tarefas podem ser executadas em um determinado momento. Sempre que uma tarefa T sofre uma transição de estado, o Escalonador reavalia as regras de dependência das tarefas relacionadas a T. Quando as condições de uma regra de dependência são satisfeitas o estado da tarefa é alterado para READY, indicando que ela está pronta para executar.

Despachante (*Dispatcher*). Prepara as tarefas para execução, fornecendo as condições necessárias para que elas possam ser executadas. O Despachante verifica periodicamente a *tasklist* à procura de tarefas que estejam prontas para executar (estado READY). Quando encontra uma tarefa nesse estado, executa uma ação conforme o tipo da tarefa: para as automáticas, cria um Gerenciador de Tarefa para controlar a execução da tarefa; para as semi-automáticas ou manuais, notifica os usuários pertencentes ao papel da tarefa a respeito de sua disponibilidade.

No caso das tarefas semi-automáticas e manuais, a escolha dos usuários a serem notificados se dá de acordo com uma ***Política de Notificação*** (escolhida no momento da criação da instância e modificável durante sua execução):

1. Todos os usuários do papel;
2. Os n usuários do papel com menos tarefas atribuídas;
3. Os n usuários do papel com menos tarefas selecionadas;
4. Os n usuários do papel que se conectam ao SGWF com mais freqüência;
5. Os n usuários do papel que se conectaram mais recentemente ao SGWF.

É também responsabilidade do PM tratar eventuais falhas nos Gerenciadores de Tarefa. Nestes casos, o PM pode criar um novo TM, na mesma máquina ou em outra, inicializando-o com o último estado conhecido do TM que falhou.

3.2.5 Gerenciador de Tarefa

A função do Gerenciador de Tarefa (*Task Manager — TM*) é controlar a execução de uma tarefa automática. O TM é criado por um PM e inicializado com a *task definition*, a partir da qual invoca a aplicação responsável pela execução da tarefa, passando os dados relativos à tarefa.

Após iniciar a execução da tarefa o TM a monitora e, quando ela é finalizada, notifica o PM que o criou, informando o tipo de término (com sucesso ou com falha). Se a execução de uma tarefa falha, o TM pode executar a tarefa novamente até obter sucesso ou até alcançar um limite de tentativas definido na especificação da tarefa.

3.2.6 Gerenciador de Lista de Trabalho

O Gerenciador de Lista de Trabalho (*Worklist Manager — WM*) tem a função de controlar e manter uma lista com as tarefas que podem ser executadas por um determinado usuário. Essa lista recebe o nome de ***Lista de Trabalho*** (*Worklist*) e as tarefas são denominadas ***Itens de Trabalho*** (*Workitems*).

Cada *workitem* corresponde a uma tarefa de um processo. Dessa forma, sempre que o estado de um *workitem* sofre alteração, o estado da tarefa correspondente também é atualizado. Essa relação garante que as alterações efetuadas pelos usuários sobre os *workitems* se refletem nos processos. Os *workitems* correspondem sempre a tarefas semi-automáticas ou manuais, já que as automáticas são tratadas de forma distinta (através de TMs) e não são adicionadas à *worklist*.

Existe um e somente um WM para cada usuário do sistema, executado sempre na máquina de seu usuário correspondente.

3.3 Funcionamento

Esta seção descreve o funcionamento do WorkToDo, ressaltando os aspectos relativos à operação no ambiente de comunicação sem fio.

3.3.1 Interação dos Usuários com o Sistema

Um usuário interage com o SGWF por meio de uma interface que fornece mecanismos de comunicação com os diversos componentes do sistema. A interação ocorre principalmente através da *worklist*, pela qual o usuário pode efetuar as seguintes ações:

Visualizar *workitems*. A *worklist* mostra todos os *workitems* do usuário, listados de acordo com um de quatro tipos de ordenamento: por ordem de chegada, por prioridade, por prazo e por tamanho dos dados relacionados. O usuário também pode mover *workitems* para a posição que desejar.

Selecionar *workitems*. Quando o usuário deseja executar um *workitem* ele deve selecioná-lo para execução. Nesse momento o PM correspondente é notificado e remove o *workitem* da *worklist* dos demais usuários daquele papel, avisando seus respectivos WMs. O usuário recebe então uma confirmação e a partir daí pode executar o *workitem*. Esse processo de seleção evita que dois usuários executem um mesmo *workitem*.

Trancar *workitems*. Quando o usuário deseja executar um *workitem* em modo desconectado ele deve trancá-lo[1] (como será visto na Seção 3.3.4). Dessa forma, o *workitem* será removido das *worklists* dos demais usuários daquele papel, exatamente como se o usuário o tivesse selecionado. Um *workitem* somente pode ser trancado se sua tarefa correspondente foi definida como sendo passível de operação desconectada.

[1] Do termo em inglês *lock*.

Além disso, um usuário pode também verificar quais são os processos em execução e consultar o estado dos processos em que ele participa. Ainda lhe é permitido criar instâncias de um determinado tipo de processo, desde que ele pertença ao papel criador daquele tipo.

Para o SGWF, os usuários podem estar ***ativos*** ou ***inativos***. Um usuário ativo está conectado ao SGWF, podendo receber notificações de tarefas prontas para executar, selecionar tarefas para execução, criar processos e consultar estados de processos, entre outras operações. Um usuário inativo não está conectado ao SGWF, portanto não pode interagir com componentes do sistema que exijam chamadas remotas, limitando-se às interações com o Gerenciador de Lista de Trabalho, conforme será visto na Seção 3.3.4.

O WorkToDo define ainda dois tipos especiais de usuários:

Administrador. Possui alguns privilégios em relação aos demais usuários, tanto de ordem administrativa quanto gerencial. É capaz de consultar e alterar informações a respeito de usuários, papéis, tipos de processo, modelos de tarefas e aplicações. O administrador é na verdade um papel, que pode ser assumido por diversos usuários.

Responsável pelo Processo. Toda instância de processo possui um usuário responsável, que deve tomar certas decisões quando necessário. Para tanto, o sistema notifica o usuário a respeito de prazos de tarefas, de tarefas que falharam em sua execução, bem como do início e término da execução do processo.

3.3.2 Prazos para Tarefas

As tarefas podem possuir um prazo de execução, especificado em sua definição, que representa o tempo máximo para que uma tarefa seja executada. Caso o prazo de uma tarefa seja atingido, o responsável pelo processo decide se a execução do processo deve ou não ser interrompida.

De tempos em tempos (quando restam 48, 24, 12 e 6 horas de prazo), o SGWF informa os usuários da aproximação do prazo. Além disso, quando o valor do prazo é igual a 24 horas, a execução da tarefa é atribuída automaticamente ao responsável pelo processo, mesmo que ela tenha sido selecionada por algum usuário (nesse caso o *workitem* correspondente à tarefa é removido da *worklist* do usuário). Isso evita que uma tarefa não seja executada dentro de seu prazo estipulado por negligência ou descuido dos usuários.

O controle dos prazos das tarefas é efetuado pelos PMs aos quais elas pertencem. Entretanto, se um usuário tranca uma tarefa e torna-se inativo, não receberá as notificações enviadas pelo PM. Assim, quando um usuário está inativo o WM também exerce controle sobre os prazos das tarefas presentes na *worklist* do usuário, efetuando as notificações no lugar do PM. A transferência de tarefas para o responsável continua sendo efetuada normalmente pelo PM.

3.3.3 Prefetching

A técnica de *prefetching* consiste em copiar para a máquina do usuário um *workitem* que mais tarde pode vir a ser executado de forma desconectada (Seção 3.3.4). Assim, quando esse *workitem* for trancado, as informações necessárias para sua execução já estarão disponíveis antes mesmo de sua seleção, minimizando o tempo de espera. O *prefetching* é sempre efetuado em *background*, de forma transparente para o usuário, aproveitando a largura de banda disponível.

O *prefetching* ocorre da seguinte forma: enquanto o usuário está conectado ao SGWF, o WM verifica periodicamente se a *worklist* contém algum *workitem* que pode ser executado de

forma desconectada. Em caso afirmativo, o WM copia o *workitem* para a máquina do usuário. Copiar um *workitem* significa copiar tanto os dados quanto a aplicação relacionados a ele (se existirem). Quando a cópia é completada, o *workitem* é dito ***transferido***.

Todos os *workitems* que podem ser executados de forma desconectada são incluídos no *prefetching* e a ordem com que são copiados é dada de acordo com seu ordenamento na *worklist*. Entretanto, um *workitem* trancado ganha prioridade sobre os demais.

Podem ocorrer casos em que os dados copiados jamais são utilizados, pois o *workitem* correspondente não é trancado pelo usuário. Nesses casos os dados ficariam ocupando espaço na máquina do usuário sem motivo. Por isso, quando um *workitem* transferido é removido da *worklist* do usuário, os dados relativos ao *workitem* são também removidos.

3.3.4 Operação Desconectada

No WorkToDo, um usuário participante de um processo pode executar tarefas sem estar conectado ao SGWF. O usuário escolhe um ou mais *workitems* de sua *worklist*, desconecta-se do sistema, e após executar os *workitems* (minutos, horas ou dias mais tarde), conecta-se novamente ao sistema e informa os resultados de seu trabalho.

O WorkToDo oferece suporte à operação desconectada planejada. Isto significa que quando o usuário pretende trabalhar de forma desconectada ele notifica o sistema, que toma então as medidas necessárias. A estratégia adotada para efetuar essa notificação foi a de ***trancamento*** de tarefas, na qual o usuário reserva *workitems* para executar em modo desconectado. Durante a operação desconectada o usuário pode visualizar e modificar sua *worklist* normalmente, além de executar *workitems* da mesma forma como quando está conectado. A única diferença é que somente aparecem na *worklist* aqueles *workitems* que foram trancados antes da desconexão; os demais ficam desabilitados.

Para poder executar um *workitem* em modo desconectado é necessário que os dados relativos ao *workitem* estejam disponíveis localmente. Por isso, antes da desconexão esses dados devem ser copiados para a máquina do usuário. Essa cópia é efetuada no momento da desconexão do usuário (ou antecipadamente, se considerarmos o *prefetching*). Além disso, quando o usuário se reconecta, o SGWF deve ser informado a respeito do resultado dos *workitems* que foram executados. O WorkToDo define então dois protocolos, um para a desconexão e outro para a reconexão dos usuários.

Protocolo de Desconexão

1. O usuário notifica seu WM de que deseja se desconectar do SGWF, estabelecendo o tempo máximo que aceita esperar até a desconexão;
2. O WM verifica se o usuário possui algum *workitem* selecionado para execução. Se possui, não permite a desconexão;
3. O WM notifica o URM de que o usuário está se desconectando;
4. O URM marca o usuário como inativo;
5. O WM calcula o tamanho de cada *workitem* trancado e não transferido, definido como a soma do tamanho da aplicação e dos dados relacionados ao *workitem* (se existirem). A aplicação somente é considerada caso ela não esteja instalada na máquina do usuário;
6. O WM calcula o volume total de dados a transferir, definido como a soma dos tamanhos dos *workitems* trancados e não transferidos;

7. O WM verifica se a largura de banda da conexão permite que o volume total de dados seja transferido dentro do tempo estabelecido pelo usuário. Em caso negativo, o sistema indica ao usuário o tempo necessário e permite que o usuário aceite esse tempo ou destranque algum *workitem*, retornando a seguir ao item 5;
8. O WM verifica se há espaço na máquina do usuário para abrigar o volume total de dados. Em caso negativo, o sistema notifica o usuário para que este solucione o problema (liberando espaço em disco, por exemplo) ou destranque algum *workitem*, retornando a seguir ao item 5;
9. O WM copia cada *workitem* trancado e não transferido;
10. O WM desabilita todos os *workitems* que não estão trancados;
11. O SGWF notifica o usuário, através da interface, de que ele está desconectado do sistema.

O item 2 garante que um usuário não irá se desconectar com algum *workitem* selecionado. Isto é necessário visto que, durante a execução de um *workitem*, podem ser necessários dados remotos que não estarão disponíveis se o usuário estiver desconectado.

Como se trata de um ambiente de comunicação sem fio, a qualquer momento durante a execução desse protocolo podem ocorrer quedas de conexão, tanto por falhas como por decisão do usuário. Nesse caso, quando o usuário reconecta a transferência dos dados é retomada a partir do ponto onde foi interrompida.

Além disso, um *workitem* transferido pode ser executado em modo desconectado mesmo se a execução do protocolo de desconexão for interrompida, pois ele já foi trancado (evitando que outros usuários o executem) e todos os dados que ele necessita estão na máquina do usuário. Dessa forma, é possível que um usuário trabalhe de forma desconectada mesmo na ocorrência de desconexões não planejadas.

Protocolo de Reconexão

1. O usuário notifica seu respectivo WM de que deseja se conectar ao SGWF;
2. O WM notifica o URM de que o usuário está se conectando;
3. O URM marca o usuário como ativo e atualiza seu endereço IP;
4. O SGWF notifica o usuário, através da interface, de que ele está conectado e de que os resultados da execução dos *workitems* serão transferidos para o SGWF;
5. O WM habilita todos os *workitems*;
6. Para cada *workitem* executado em modo desconectado, o WM:
   (a) Transfere em *background* os dados resultantes da execução ao PM correspondente;
   (b) Remove o *workitem* da *worklist*.

No momento da desconexão, os *workitems* que não estavam trancados são desabilitados; por isso o item 5 habilita-os novamente. Podem existir *workitems* desatualizados, ou seja, que já foram selecionados, trancados ou mesmo executados por outros usuários. Por isso, na próxima interação entre WM e PM a *worklist* é atualizada e, nesse momento, alguns *workitems* podem ser removidos. Se antes dessa atualização o usuário tentar selecionar um *workitem* inválido o PM o notificará de que o *workitem* não está mais disponível. Como no protocolo anterior, podem ocorrer desconexões durante a execução do protocolo e, da mesma forma, quando o usuário reconecta a transferência dos dados é retomada do ponto onde foi interrompida.

É importante ressaltar que, mesmo em um SGWF que suporta operação desconectada, nem todas as tarefas devem ser executadas em modo desconectado [3]. Deve-se levar em consideração fatores como tamanho de dados e aplicações e atraso na execução de tarefas e processos (deviso à demora inerente da operação desconectada). Faz-se necessária então uma análise da relação custo-benefício de tornar as tarefas passíveis de execução desconectada, sob pena de degradar o desempenho do sistema e não obter os resultados esperados.

3.3.5 Operação Semi-Conectada

A operação semi-conectada ocorre quando um usuário acessa o sistema através de uma conexão sem fio (por exemplo, via modem). O usuário executa as tarefas da mesma forma como se estivesse em uma conexão convencional, com a diferença de que a conexão é instável e possivelmente com baixa largura de banda, tornando falhas mais freqüentes.

No caso de queda na conexão, o WM desabilita tanto os *workitems* que não estão trancados quanto os que estão trancados mas não foram transferidos, permanecendo habilitados somente aqueles *workitems* trancados e transferidos. Dessa forma o usuário pode continuar trabalhando normalmente nesses *workitems*, da mesma forma como na desconexão planejada.

Se o usuário tinha algum *workitem* selecionado no momento da queda de conexão, existem dois casos a serem considerados. Se o *workitem* utiliza dados remotos, o usuário não poderá acessá-los enquanto não reconectar, então a execução da tarefa é temporariamente paralisada. Quando o usuário reconecta, os dados se tornam novamente acessíveis e a execução pode ser retomada. Se o *workitem* utiliza uma aplicação remota, a execução da aplicação será terminada. Cabe ao usuário, após reconectar, reiniciar a aplicação e retomar a execução da tarefa.

Durante a operação semi-conectada podem existir períodos nos quais há pouca ou nenhuma comunicação entre o usuário e o SGWF. Esses períodos são utilizados para *prefetching*, como explicado anteriormente, aproveitando o máximo possível da largura de banda da conexão.

4 Implementação

Esta seção contém um exemplo de processo, além dos resultados obtidos com os testes do protótipo do WorkToDo. O protótipo foi implementado em Java (JDK 1.2) e para fornecer suporte à distribuição foi utilizado o OrbixWeb 3.1c [10], a implementação de CORBA da IONA [6]. Os componentes da arquitetura foram todos implementados como servidores CORBA.

4.1 Exemplo

Para demonstrar a utilização do WorkToDo, vamos considerar o exemplo descrito anteriormente, onde uma empresa que presta serviços de manutenção cria um processo para gerenciar seus serviços. A especificação de tal processo na Linguagem de Definição de Processo do WorkToDo é a seguinte (as demais tarefas foram omitidas por motivos de espaço):

```
WORKFLOW Manutencao {
     FILE OrdemServico {
          NAME "/processos/arquivos/OrdemServico.doc";
          SIZE 100;
```

```
        }
        ⋮
        TASK VisitarCliente:PreencherFormulario {
             ROLE Tecnico;
             IN_CONTEXT OrdemServico;
             OUT_CONTEXT OrdemServico;
             DEPENDS CriarOrdemServico -> SUCCEEDED;
             DESCRIPTION "Preencher os campos 'Serviços' e
                          'Materiais' da Ordem de Serviço";
        }
        ⋮
    }
```

O processo, denominado `Manutencao`, contém a tarefa `VisitarCliente`, instância do modelo de tarefa `PreencherFormulario`. Essa tarefa, de responsabilidade dos usuários do papel `Tecnico`, somente será iniciada se a tarefa `CriarOrdemServico` for executada com sucesso. O único dado de entrada e saída de `VisitarCliente` é o arquivo `OrdemServico.doc`, identificado por `OrdemServico`, localizado no diretório `/processos/arquivos/` e de tamanho igual a `100Kb`. A especificação contém ainda uma descrição de como proceder para executar a tarefa.

O modelo de tarefa `PreencherFormulario` é definido da seguinte forma:

```
TASK PreencherFormulario {
     TYPE Semi-automatic;
     APPLICATION EditorTexto;
     PRIORITY 10;
     DEADLINE 48 HOURS;
     DISCONNECTED_OPERATION true;
}
```

A tarefa é do tipo semi-automática e a aplicação relacionada a ela é `EditorTexto` (cujas características estão definidas separadamente). São ainda indicados a prioridade e o prazo para execução da tarefa. A cláusula `DISCONNECTED_OPERATION` indica se a tarefa pode ser executada em modo desconectado (`true`) ou não (`false`).

Supondo que existe uma instância do processo `Manutencao` em execução e que a tarefa `CriarOrdemServiço` foi executada com sucesso, a tarefa `VisitarCliente` já pode ser executada, pois suas dependências foram satisfeitas. Como se trata de uma tarefa semi-automática, o Despachante adiciona um *workitem* correspondente à tarefa na *worklist* de todos os usuários do papel `Tecnico` (ou de alguns desses usuários, conforme a política de notificação escolhida).

A Figura 4 mostra a *worklist* de um usuário do papel `Tecnico`, no caso de existirem várias instâncias do processo `Manutencao` em execução, bem como instâncias de um outro processo `wf2`. A *worklist* pertence ao usuário `Paulo`, que no momento está conectado ao SGWF, e contém cinco *workitems*, sendo três deles correspondentes a tarefas `VisitarCliente` de diferentes instâncias do processo `Manutencao` e os outros dois correspondentes a tarefas `task_1` pertencentes a instâncias de `wf2`. O primeiro *workitem* está pronto para ser

executado (estado READY), o segundo e o terceiro estão selecionados (SELECTED) e o quarto está trancado (LOCKED). Tarefas de mesmo nome são diferenciadas entre si pelo nome da instância à qual pertencem.

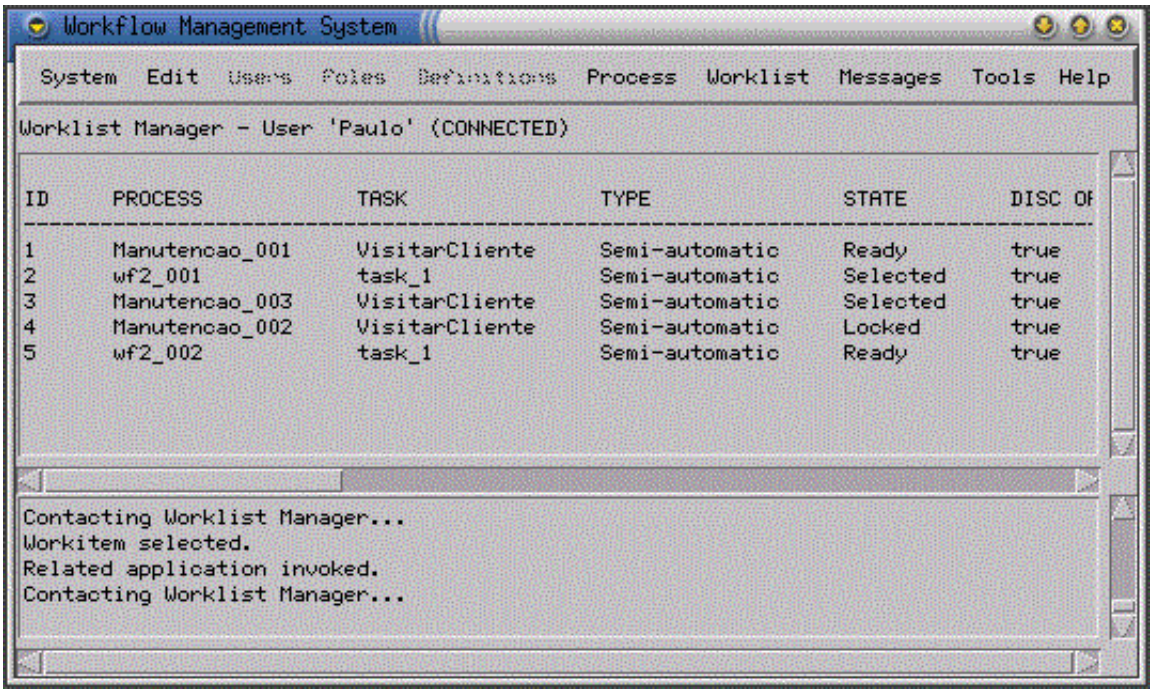

Figura 4: Exemplo de Worklist.

A *worklist* contém diversas informações a respeito dos *workitems*, como o nome da instância de processo e da tarefa correspondentes, o tipo do *workitem*, seu estado atual e se pode ser executado em modo desconectado. A partir dos menus o usuário pode interagir com a *worklist*, com os *workitems* e com o SGWF como um todo.

4.2 Resultados

Para efetuar os testes do protótipo desenvolvido, o URM, o DM e o IM foram executados cada um em uma máquina diferente. A máquina de cada PM era escolhida no momento da criação da instância; as máquinas dos TMs seguiam as definições das aplicações (o TM é criado na máquina onde a aplicação está disponível); as máquinas dos WMs eram as mesmas de seus respectivos usuários.

As operações desconectada e semi-conectada mostraram-se possíveis de serem executadas e o sistema se mostrou estável frente a quedas de conexão, sem gerar inconsistências. Quando ocorre uma queda de conexão, o usuário não pode notificar o SGWF de que está se desconectando, por isso, se algum PM não consegue contactar determinado WM um certo número de vezes, o usuário correspondente ao WM é considerado inativo. Quando esse WM interage com algum outro componente, o usuário é novamente considerado ativo.

Com a utilização do *prefetching* o tempo de execução do protocolo de desconexão mostrou-se bastante pequeno, reduzindo-se praticamente ao tempo da chamada remota ao URM. Isto porque a transferência dos *workitems* trancados, que é o item do protocolo que consome mais tempo, já foi executado.

5 Conclusão

Neste artigo apresentamos o WorkToDo, um SGWF para ambientes de comunicação sem fio. Com esse sistema, o usuário pode executar tarefas a partir de seu *laptop* ou computador pessoal, sem que hajam restrições de mobilidade. Além disso o usuário pode executar tarefas sem estar conectado ao sistema, bastando para isso indicar, através de um sistema de trancamento, as tarefas que deseja executar em modo desconectado. No momento da desconexão o WorkToDo executa um protocolo que copia todos os dados referentes a essas tarefas para a máquina do usuário. A execução das tarefas se dá sobre esses dados copiados. Na reconexão é executado outro protocolo, que transfere os resultados da execução para o WorkToDo.

Consideramos também aspectos como o tratamento local de prazos de tarefas (efetuado pelo WM) e a cópia de dados e de resultados (efetuados pelo WM na execução dos protocolos de conexão e desconexão).

O WorkToDo ainda implementa o *prefetching* para otimizar a cópia de *workitems* e aproveitar a largura de banda da conexão. Enquanto o usuário está conectado ao SGWF, possivelmente executando tarefas, o WM copia em *background* os *workitems* que podem ser executados de forma desconectada. No caso do usuário trancar esses *workitems*, eles já estarão disponíveis localmente.

Referências

[1] G. Alonso, D. Agrawal, A. El Abbadi, R. Günthör, M. Kamath, and C. Mohan. Exotica/FMDC: Handling Disconnected Clients in a Workflow Management System. In *Proceedings of the 3rd International Conference on Cooperative Information Systems*, pages 99–110. Vienna, Austria, 1995.

[2] D. Barbará, S. Mehrotra, and M. Rusinkiewicz. INCAS: A Computation Model for Dynamic Workflows in Autonomous Distributed Environments. Technical report, Department of Computer Science, University of Houston, 1994.

[3] C. Büssler. User Mobility in Workflow Management Systems. In *Proceedings of the Telecommunication Information Networking Architecture Conference (TINA95)*. Melbourne, Australia, 1995.

[4] J. Eder, H. Groiss, and W. Liebhart. Workflow Management and Databases. In *Proceedings of the 2nd Forum International d'Informatique Appliquee.* Tunis, 1996.

[5] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.

[6] IONA Technologies - http://www.iona.com, Fevereiro de 2002.

[7] S. Jablonski. Functional and Behavioral Aspects of Process Modelling in Workflow Systems. In A. Benczur G. Chroust, editor, *CON 94 Workflow Management: Challenges, Paradigms and Products*, pages 113–133. R. Oldenburg, 1994.

[8] J. Jing, K. Huff, H. Sinha, B. Hurtwitz, and B. Robinson. Workflow and Application Adaptations in Mobile Environments. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications.* New Orleans, Louisiana, USA, 1999.

[9] M. Kamath and K. Ramamritham. Bridging the Gap Between Transaction Management and Workflow Management. In *Proceedings of NSF Workshop on Workflow and Process Automation in Information Systems.* Athens, Georgia, 1996.

[10] OrbixWeb Programmers's Guide. IONA Technologies PLC. http://www.iona.com, Fevereiro de 2002.

[11] J. Veijalainen, A. Lehtola, and O. Pihlajamaa. Research Issues in Workflow Systems. Technical report, VTT Information Technology, Finland, 1995.