APOIO À INTEGRAÇÃO DE PROCESSOS DE NEGÓCIOS EM EMPRESAS VIRTUAIS

Marcelo Trannin Machado

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

Aprovada por:

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

Prof. Maria Luiza Machado Campos, Ph.D.

Prof. José Roberto de Souza Blaschek, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2002

MACHADO, MARCELO TRANNIN

Apoio à Integração de Processos de Negócios em Empresas Virtuais [Rio de Janeiro] 2002

X, 130 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Empresas Virtuais; 2. Processos de Negócios; 3. Workflow; 4. Comércio Eletrônico

I. COPPE/UFRJ II. Título (série)

A Juvencio

Agradecimentos

Ao Prof. Marcos Borges, pelo profissionalismo, competência e otimismo com que me orientou durante o mestrado e, particularmente, nessa tese. Aprendi muito com sua experiência e exemplo.

À Profa. Maria Luiza. Foi uma honra e um prazer "reencontrá-la" ao fim do mestrado, após os anos de orientação durante a graduação.

Ao Prof. Blaschek, por ter aceito prontamente participar dessa banca de tese e pela compreensão com as mudanças de datas que ocorreram.

Aos meus pais e irmãos, pelo apoio e também pelos sacrifícios que fizeram para que eu pudesse concluir esse trabalho.

Às amigas Flávia e Renata, companheiras nessa jornada acadêmica e na vida.

Aos amigos Márcio Dias e "Jacaré", que além de oferecerem sua amizade, ajudaram-me em meu crescimento profissional.

Ao amigo José Augusto, pela dedicação e companheirismo.

Ao pessoal da Software Design. No Rio, ao Cláudio, Marco, Carol, Luciana, Paulo, Lívia, Cátia, Maurílio e André, pelas trocas de idéias e pelo apoio. Em Campinas, ao Junqueira, pelas licenças e cópias dos aplicativos da IBM e ao Renato, pela compreensão com esse colaborador "virtual".

À amiga Cris, que talvez tenha sido a pessoa que mais se preocupou com essa tese, depois de mim...

Aos professores do PESC e de Engenharia de Software. Em especial, aos Profs. Cláudia Werner e Xexéo, pela orientação nos trabalhos e publicações realizados durante o curso.

Ao pessoal do grupo CHORD.

À amiga Laura, pela força, otimismo e bom-humor.

À Eliani, que foi uma "co-orientadora" nesse trabalho, por ter acreditado tanto em mim. Sem dúvida, não teria chegado ao fim, se não fosse seu encorajamento, sua visão e conselhos.

A todo o pessoal do grupo de Segunda, por terem me "aturado" falar tanto de tese. Em especial, ao amigo Ricardo, por ter estado ao meu lado nesse momento tão importante. Também à Celina, Mariza, Priscila e Priscila Capitã, por terem apoiado e participado.

Às Dras. Ana Marta e Ana Beatriz, pelos seus atenciosos cuidados profissionais durante esse período de tese.

Ao pessoal técnico-administrativo do PESC e do NCE.

Ao CNPq, pela bolsa concedida.

Ao meu avô Juvencio, por ter me incentivado o gosto pela leitura e pelo conhecimento.

A Deus, por sua infinita bondade e justiça: insistiu em mostrar-me o caminho todas as vezes que me senti perdido e desesperançado. Também a todos os demais que me inspiraram, apoiaram, acalmaram e deram a força necessária...

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

APOIO À INTEGRAÇÃO DE PROCESSOS DE NEGÓCIOS EM EMPRESAS VIRTUAIS

Marcelo Trannin Machado

Abril / 2002

Orientador: Marcos Roberto da Silva Borges

Programa: Engenharia de Sistemas e Computação

Este trabalho propõe o desenvolvimento de uma infra-estrutura de apoio à definição e à encenação de processos de negócios integrados em empresas virtuais, baseada nos aspectos essenciais deste tipo de processo. A interação entre as empresas é realizada tendo como base o conceito de serviços como forma de abstração de processos. A infra-estrutura é constituída de componentes que apóiam o monitoramento e a coordenação dos serviços e processos das empresas virtuais, além de ferramentas acessórias para sua modelagem e encenação. Um protótipo foi desenvolvido, visando validar os conceitos propostos.

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

SUPPORT FOR BUSINESS PROCESSES INTEGRATION IN VIRTUAL ENTERPRISES

Marcelo Trannin Machado

April / 2002

Advisor: Marcos Roberto da Silva Borges

Department: System and Computer Engineering

This work proposes an infrastructure for the support of business processes definition and enactment in virtual enterprises, based on the main aspects of this kind of process. The interaction between the enterprises is achieved by employing the concept of services, used as an abstraction of processes. The infrastructure consists of components that support the monitoring and the coordination of services and virtual enterprises' processes, besides providing modeling and enactment tools. A prototype was implemented to validate the proposed concepts.

Sumário

1 - Introdução

1.1. Motivação

Novos requisitos de mercado, como globalização e concorrência acirrada, estão obrigando as empresas a serem mais ágeis e flexíveis e procurarem novas formas de satisfazer seus clientes. Entre as estratégias adotadas estão a oferta de produtos ou serviços cada vez mais elaborados e que atendam às necessidades específicas de cada cliente, diminuição do tempo para pôr um produto no mercado (*time to market*) e diminuição do tempo de entrega de um produto ou serviço ao cliente. Para pôr um produto no mercado em menos tempo e conseguir entregá-lo rapidamente a seus clientes, relacionamentos com fornecedores e distribuidores precisam ser estreitados. De modo a oferecer produtos com maior qualidade e diferenciação em relação a seus concorrentes, as empresas estão focando seus esforços no que é exclusivamente próprio do seu negócio (*core business*) e delegando a parceiros partes de seus processos de produção. Cada parceiro, por sua vez, tem condições de agregar mais valor ao produto/serviço (ou parte dele) realizado.

A integração de empresas com outras empresas parceiras para a produção de determinados produtos ou serviços em comum representa o conceito de empresa virtual. Empresas virtuais podem ser criadas para atender a uma oportunidade específica surgida no mercado e desmanchadas ao atendê-la, após um curto período de tempo, ou para produzir bens ou serviços numa base mais perene. Outra possibilidade é a escolha dinâmica de parceiros, de acordo com critérios que variam segundo as exigências do mercado no momento (tempo de entrega, preço, qualidade, etc.).

Para que uma empresa virtual possa operar adequadamente e atingir os seus objetivos, um processo global precisa ser definido e orquestrado. Esse processo, chamado processo virtual, é composto pelos processos individuais de cada empresa constituinte da empresa virtual.

1.2. Caracterização do Problema

Ao se integrar processos diversos, de diferentes empresas, para compor um único processo virtual, diversos problemas são encontrados, de ordem tecnológica e da própria natureza dos processos. Em geral, esses problemas originam-se do seguinte dilema: como permitir que uma empresa utilize e controle os serviços oferecidos pelos processos de outra empresa, porém sem interferir na privacidade e autonomia desses processos?

Utilizando a tecnologia atual de *workflow*, uma organização precisa tornar "abertos" seus processos de negócios para permitir que outras empresas utilizem e controlem os serviços oferecidos por ela. Isso se deve ao fato de que as soluções atuais só contemplam a integração quanto à interoperabilidade de máquinas de *workflow*, no nível das atividades dos processos. Para utilizar o serviço oferecido por um processo de outra empresa, é preciso conhecer e referenciar as atividades que o compõe.

No entanto, essa abordagem deixa a organização muito exposta ao mundo externo (parceiros, clientes e concorrentes), que passa a poder observar suas estratégias de produção e também suas fraquezas. Além disso, ela precisa estar constantemente efetuando mudanças e evoluções em seus processos, a fim de melhor adaptar-se às oportunidades surgidas no mercado. Contudo, no momento em que permitem que as atividades de seus processos sejam referenciadas diretamente por outras empresas, ficam impossibilitadas de alterá-las, sob a pena de atrapalhar ou impedir o monitoramento e controle dessas empresas.

1.3. Hipótese

Apesar da limitação tecnológica atual, para se utilizar os serviços oferecidos pelos processos de negócios de uma determinada empresa não é preciso saber os detalhes de seu funcionamento interno, mas somente sua interface, isto é, como os serviços devem ser invocados, como podem ser monitorados e controlados, quais são seus critérios de qualidade e, finalmente, que resultados podem ser esperados. Logo, o nível de interface de serviço é uma abstração dos detalhes internos de um processo gerenciado por um sistema de *workflow*.

O trabalho elaborado nessa dissertação parte da seguinte hipótese: é possível integrar processos de distintas organizações no nível de interface de serviço e evitar a

interferência na privacidade e autonomia desses processos, sendo este um meio claro e bem estabelecido de monitoramento e controle sobre os processos contratados.

1.4. Enfoque de Solução

O enfoque de solução adotado nesse trabalho propõe a criação de uma camada de gerenciamento de serviços sobre os WfMS comerciais. A integração entre as organizações é feita no nível dessa camada de serviços, sem ser preciso conhecer a camada abaixo dela.

Como são abstrações dos processos de negócios de uma organização, os serviços precisam ser mapeados para os processos que os implementam no sistema gerenciador de *workflow* (WfMS). O mapeamento é feito monitorando-se os estados das atividades do processo interno. A determinadas mudanças nas atividades do processo interno correspondem mudanças no estado do serviço, que pode ser obtido pela empresa contratante. Por outro lado, operações podem ser invocadas sobre os serviços, o que causa uma ação sobre os processos internos.

As organizações possuem diversos processos definidos e sendo encenados em seus sistemas de *workflow*. A solução proposta considera esse fato e é independente de produtos comerciais específicos. Ou seja, não é preciso remodelar esses processos num novo sistema e qualquer WfMS poderá ser utilizado.

A criação de um processo virtual é feita compondo-se diversos serviços, de empresas distintas. As interações entre eles também são determinadas no processo virtual. A solução leva em conta a necessidade de definição dos processos virtuais, assim como seu gerenciamento.

Para a realização da comunicação entre as empresas, o enfoque é o uso de protocolos neutros. Nesse sentido, o uso da tecnologia de *web services* é bastante propício. Um *web service* é uma interface que descreve uma coleção de operações que são acessíveis através de mensagens padronizadas em XML (IBM-WS, 2002).

1.5. Objetivos da Tese

O objetivo desta tese é elaborar uma solução tecnológica que permita a integração de empresas no nível de serviços, a partir da abstração dos processos internos da organização que são encenados em seus sistemas gerenciadores de *workflow*.

Para se propor tal solução, uma análise dos problemas resultantes desse tipo de tarefa deve ser feita. Tal análise possui o propósito de destacar aspectos e requisitos fundamentais para uma solução satisfatória para esses problemas.

Partindo desse estudo, um ambiente para desenvolvimento de aplicações de gerenciamento de processos virtuais será especificado e um protótipo será implementado para validar a viabilidade de sua construção.

Espera-se permitir o uso da abstração de serviço para se alcançar efetivamente a privacidade e autonomia de processos, sejam eles definidos e executados em quaisquer WfMSs. Isto é, a integração será realizada entre as empresas, mas estas terão total liberdade para alterarem seus processos, mantê-los em sigilo ou mudá-los de sistema.

O ambiente deverá apoiar a definição de processos virtuais, assim como realizar o seu gerenciamento e coordenação, através da execução de serviços.

1.6. Organização da Tese

O Capítulo 2 apresenta duas das propostas mais importantes de padronização na área de *workflow*, feitas por duas entidades de grande aceitação na área, a *Workflow Management Coalition* (WfMC), e a OMG (*Object Management Group*). É apresentada também uma proposta de padrão de comércio eletrônico entre empresas, chamado ebXML.

No Capítulo 3, descreve-se um estudo sobre os problemas encontrados ao se integrar processos entre diferentes organizações. Os problemas, em geral, baseiam-se no dilema descrito na Seção 1.2.

Com o intuito de resolver parte dos problemas levantados, o Capítulo 4 apresenta o enfoque da solução proposta por esse trabalho, apontando seus requisitos e aspectos mais relevantes. Os conceitos mais importantes por trás da solução são a abstração de processos em **serviços**, através do estabelecimento de **contratos**, e acessíveis via *web* (usando a tecnologia de ***web services***)

O conjunto de requisitos descrito no Capítulo 4 serve como base para a elaboração do modelo de dados e da arquitetura do ambiente proposto, que são o tema do Capítulo 5. O modelo de dados procura refletir os conceitos presentes no Capítulo 4, enquanto a arquitetura visa a viabilização estrutural e tecnológica da solução proposta.

A partir daí, o Capítulo 6 apresenta o projeto do ambiente, mostrando as decisões de projeto tomadas e os diagramas de classes que o compõe.

Por fim, o Capítulo 7 compila um resumo sobre o trabalho, enumera seus resultados alcançados e contribuições, e apresenta possibilidades de trabalhos futuros.

2 – Padrões de *Workflow* e Comércio Eletrônico

Este capítulo apresenta duas das propostas mais importantes no sentido de padronização de conceitos e sistemas de gerenciamento de *workflow*. A padronização nesse domínio, sob o ponto de vista do assunto de integração de empresas, tem como benefício a quebra de algumas das barreiras que impedem a interoperabilidade entre sistemas de *workflow* de diferentes fabricantes e, consequentemente, o surgimento da possibilidade de integração de processos entre empresas distintas. Nesse sentido, é apresentado nesse capítulo também um padrão para comércio eletrônico entre empresas surgido recentemente.

2.1. *Workflow Management Coalition* (WfMC)

A *Workflow Management Coalition* (WfMC) é um grupo de empresas que se reuniram para estudar a questão de compatibilidade entre os diversos sistemas de *workflow* comerciais, uma vez identificado que todos os produtos de gerenciamento de *workflow* possuem características comuns, habilitando-os, assim, a atingirem um nível de interoperabilidade através do uso de interfaces e padrões comuns. O intuito da WfMC é, portanto, criar especificações que padronizem o uso de sistemas de gerenciamento de *workflow*.

Entre os documentos básicos criados pela WfMC estão o documento que apresenta a terminologia e glossário de sistemas gerenciadores de *workflow* (WfMSs) (WfMC-TC-1011, 1999) e o que apresenta o modelo de referência (WFMC-TC00-1003, 1995).

No modelo de referência está descrita a arquitetura definida pela WfMC para WfMSs. A Figura 2.1 apresenta os principais componentes dessa arquitetura e as respectivas interfaces para comunicação entre sistemas de *workflow* e as demais aplicações que interagem com esses sistemas.

Foram definidas cinco interfaces, cada uma cobrindo um importante aspecto de interação necessário para a implantação de sistemas de *workflow*. A seguir, cada uma delas será brevemente apresentada.

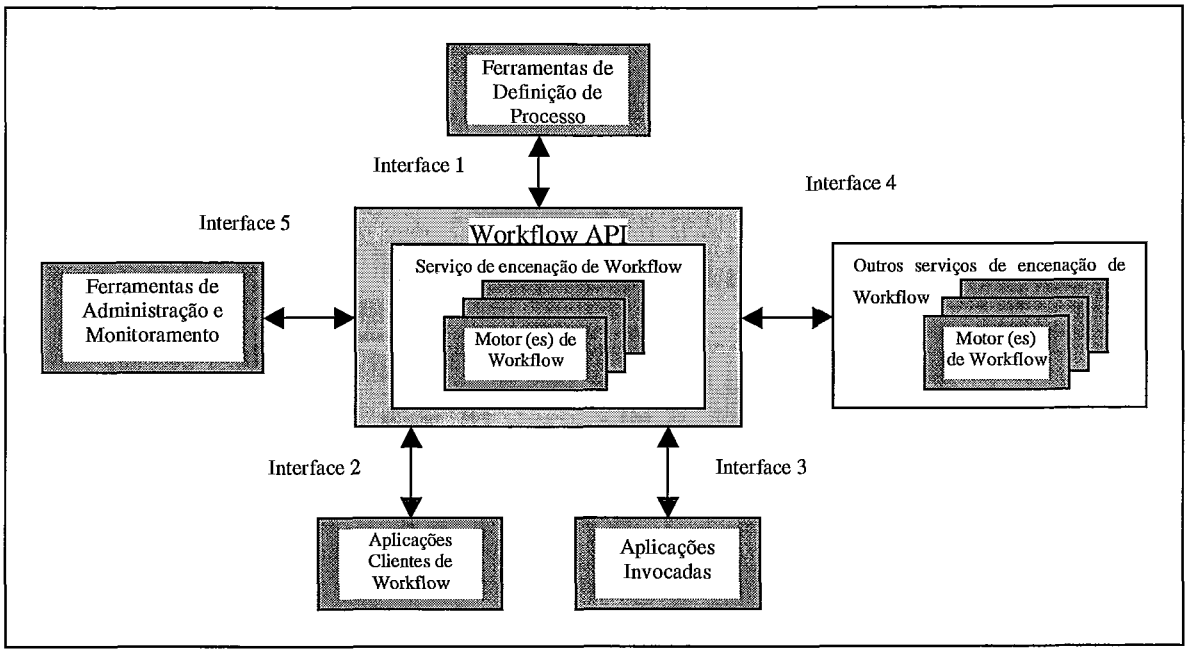

Figura 2.1 – Arquitetura de Referência da WfMC (WFMC-TC00-1003, 1995)

Interface 1 – Definição de Processo

Essa interface permite que definições de *workflow* (processos, atividades, regras de negócio, papéis e usuários, estruturas de dados, etc.) criadas em ferramentas próprias sejam introduzidas na máquina de *workflow* para encenação de processos. A linguagem padrão para definição de *workflow* estabelecida pela WfMC é a WPDL (*Workflow Process Definition Language*) (WFMC-TC-1016, 1999).

Interface 2 – Aplicações Clientes de *Workflow*

Essa interface especifica como pode ser feita a interação entre aplicações clientes e a máquina de *workflow*. Sua maior importância está em permitir que aplicações de tratamento de listas de trabalho se comuniquem com a máquina de *workflow*, permitindo assim que usuários interajam com o WfMS.

Interface 3 – Aplicações Externas Invocadas

O intuito dessa interface é definir como o sistema de *workflow* pode invocar aplicações externas que auxiliem os usuários na realização de tarefas relacionadas aos seus itens de trabalho.

Interface 4 – Interoperabilidade de Sistemas de *Workflow*

A interoperabilidade entre diversas máquinas de *workflow*, possivelmente de diferentes vendedores, para encenação de processos é definida na interface 4 da WfMC.

Interface 5 – Administração e Monitoramento

Através da interface 5 é possível administrar e monitorar desempenho de encenações de processos. Duas funções importantes dessa interface são permitir a auditoria e simulação de processos, usadas para melhoria contínua de processos.

As interfaces 2 (aplicações clientes) e 4 (interoperabilidade entre máquinas de *workflow*) serão mais detalhadas nas seções 2.1.1. e 2.1.2, respectivamente, por serem mais relevantes para esse trabalho.

Outro aspecto interessante definido pela WfMC e de grande relevância para esse trabalho diz respeito ao ciclo de vida de instâncias de processo e atividades. Esses ciclos de vida indicam os estados possíveis pelas quais esses objetos de *workflow* podem passar e a transições possíveis de um estado para outro. Esse item será coberto pela seção 2.1.3.

2.1.1. Interface 2 – *Workflow Management Application Programming Interface*

A interface 2 da WfMC (WFMC-TC-1009, 1998) define uma API (*Application Programming Interface*), chamada de WAPI, com o intuito de estabelecer uma interface padrão que permita o acesso de aplicações a sistemas de *workflow*. A WAPI estabelece os seguintes grupos de funções:

- Conexão: provê comandos para conexão e posterior desconexão ao sistema de *workflow*;
- Definição de *Workflow*: fornece funções para definição de modelos de *workflow*;
- Controle de Processo: contém funções que alteram o estado operacional de instâncias de processo;
- Controle de Atividade: contém funções que alteram o estado operacional de instâncias de atividade;

- Status de Processo: as funções deste grupo têm a intenção de fornecer uma visão do trabalho feito, trabalho a ser feito, trabalho associado com um participante ou grupo de participantes do *workflow*, etc;
- Status de Atividade: idem ao grupo "Status de Processo", porém no nível de atividades;
- Lista de Trabalho: possui funções que permitem o acesso de participantes do *workflow* a informações sobre tarefas às quais eles foram assinalados;
- Administração: provê funcionalidades necessárias para desempenhar funções de administração e manutenção de um sistema de *workflow*.

Dentre esses grupos de funções, os mais importantes para esse trabalho são: "Conexão", "Controle de Processo" e "Controle de Atividade".

A API definida pela WfMC foi criada originalmente visando sua implementação na linguagem C. Logo, essa API segue uma orientação procedural. No entanto, em um dos apêndices da especificação (WfMC-TC-1009) é definida uma forma orientada a objetos da API, descrita a partir de interfaces declaradas na linguagem IDL da OMG. Essas interfaces contêm atributos e, principalmente, as operações correspondentes às funções declaradas na API original.

As interfaces foram divididas em três módulos: um contendo interfaces genéricas, um para agrupar as interfaces relacionadas a aplicações clientes e o último contendo interfaces relativas à definição de processos.

A seguir, serão mostradas algumas das interfaces mais importantes, sendo estas também as mais relevantes para esse trabalho, agrupadas por módulo. As interfaces estão escritas na linguagem IDL.

➢ Módulo CfWorkflowFacilityBase

Interfaces:

`WorkflowObject`

A interface `WorkflowObject` é a mais importante deste módulo, pois define os atributos e operações comuns a todos os objetos de *workflow*. Essa interface servirá de base

para todas as outras interfaces que representam objetos de *workflow* e seus atributos e operações podem ser vistos na Figura 2.2.

```
interface WorkflowObject {

      attribute WMTName name;
      attribute WMTId id;
      void listValidStates (
                  in Filter filter,
                  in boolean countFlag,
                  out WMTStates states,
                  out long count);

      void changeState (in WMTState newState)
            raises (TransitionNotAllowed, InvalidState);

      void getState ( out WMTState currentState);

      void openAttributeList (
                  in Filter filter,
                  in boolean countFlag,
                  out AttributeList attributes,
                  out long count)
            raises (InvalidFilter);

      void getAttributeValue (
                  in WMTName name
                  out Attribute attribute)
            raises (InvalidAttribute);

      void assignAttribute ( in Attribute attribute)
            raises(InvalidAttribute, AttributeAssignmentFailed);

      void assignAttributes ( in Attributes attributes)
            raises (InvalidAttribute, AttributeAssignmentFailed);

};
```

Figura 2.2 – Interface WorkflowObject

Exceções:

O módulo CfWorkflowFacilityBase traz também a definição de algumas exceções que são levantadas pelas operações das interfaces definidas. A exceção `InvalidState` é levantada quando se tenta realizar uma transição para um estado não conhecido. Outra exceção presente é `TransitionNotAllowed`, usada para indicar que uma determinada transição de estados não é possível a partir do estado atual do objeto. Há ainda exceções usadas para lidar com filtros (de consultas) e atribuição de valores a atributos de objetos.

- Módulo CfWFApplicationClient

Interfaces:

`ApplicationClientServer`

A interface `ApplicationClientServer` permite que um usuário se conecte ao sistema de *workflow* (Figura 2.3).

```
interface ApplicationClientServer {

      attribute CfWFBase :: WMTName engineName;
      attribute CfWFBase :: WMTName scope;

      void connect(
                  in CfWFBase :: WMTName userId,
                  in string password)
            raises (ConnectFailed);

      void disconnect()
            raises (NotConnected);

      ProcessDefinitionList openProcessDefinitionsList(
                  in CfWFBase :: Filter filter,
                  in boolean countFlag)
            raises (InvalidFilter, NotConnected);
```

```
        ProcessInstanceList openProcessInstancesList (
                    in CfWFBase :: Filter filter,
                    in boolean countFlag)
              raises (InvalidFilter, NotConnected);

        ActivityInstanceList openActivityInstancesList (
                    in CfWFBase :: Filter filter,
                    in boolean countFlag)
              raises (InvalidFilter, NotConnected);

        WorkList openWorkList (
                    in CfWFBase :: Filter filter,
                    in boolean countFlag)
              raises (InvalidFilter, NotConnected);

        ProcessInstance getProcessInstance(
                    in CfWFBase :: WMTId processInstanceId)
              raises (InvalidId);

        ActivityInstance getActivityInstance(
                    in CfWFBase :: WMTId processInstanceId,
                    in CfWFBase :: WMTId activityInstanceId)
              raises (InvalidId);

        Workitem getWorkitem(
                    in CfWFBase :: WMTId processInstanceId,
                    in CfWFBase :: WMTId workItemId)
              raises (InvalidId);
};
```

Figura 2.3 – Interface ApplicationClientServer

`ProcessInstance`

A interface `ProcessInstance` provê operações para acessar e modificar estados e atributos de um objeto instância de processo e herda atributos e operações da interface `WorkflowObject`. Na Figura 2.4 são mostrados seus atributos e operações.

```
interface ProcessInstance : CfWFBase :: WorkflowObject{
      attribute CfWFBase :: WMTDataRef    dataReference;
      attribute long                      priority;

      ProcessDefinition getParentProcessDefinition ();

      void start ()
            raises (NotConnected, TransitionNotAllowed);

      void terminate();
            raises (NotConnected, TransitionNotAllowed);

      void abort();
            raises (NotConnected, TransitionNotAllowed);

      CfWFBase :: WMTWflParticipants listAssignedParticipants ()
            raises (NotConnected);
};
```

Figura 2.4 – Interface ProcessInstance

`ActivityInstance`

A interface `ActivityInstance` provê operações para acessar e modificar estados e atributos de um objeto instância de atividade e herda atributos e operações da interface `WorkflowObject`. Seus atributos e operações são apresentados na Figura 2.5.

```
interface ActivityInstance : CfWFBase :: WorkflowObject {

      attribute CfWFBase :: WMTDataRef dataReference;
      attribute long priority;

      ProcessInstance getParentProcessInstance ();

      CfWFBase :: WMTWflParticipants listAssignedParticipants ()
            raises (NotConnected);
};
```

Figura 2.5 – Interface ActivityInstance

WorkItem

A interface WorkItem provê operações para acessar e modificar estados e atributos de um objeto item de trabalho e herda atributos e operações da interface WorkflowObject. A Figura 2.6 apresenta suas operações e atributos.

```
interface WorkItem : CfWFBase :: WorkflowObject {
      attribute CfWFBase :: WMTDataRef dataReference;
      attribute long priority;

      ProcessInstance getParentProcessInstance ();

      ActivityInstance getParentActivityInstance ();

      void reassign (
                  in CfWFBase::WMTWflParticipant sourceUser,
                  in CfWFBase::WMTWflParticipant targetUser)
            raises (NotConnected, InvalidSourceUser, InvalidTargetUser);

      void get ()
            raises (NotConnected, TransitionNotAllowed);

      void complete()
            raises (NotConnected, TransitionNotAllowed);

      CfWFBase :: WMTWflParticipant getAssignedParticipant()
            raises (NotConnected);
};
```

Figura 2.6 – Interface WorkItem

2.1.2. Interface 4 – *Interoperability*

A interface 4 da WfMC (WFMC-TC-1012, 1999) é uma especificação abstrata que define as funcionalidades necessárias para a realização da interoperabilidade entre diferentes máquinas de *workflow*. Existem algumas especificações "concretas" (*bindings*) para essa especificação, cada uma utilizando um diferente protocolo de transporte. Os protocolos definidos até agora são: MIME (WfMC-TC-1018, 2000) e XML (Wf-XML, WfMC-TC-1023, 2001), tendo sido esse último incorporado e aprimorado pela WfMC a

partir de um outro protocolo criado anteriormente com o mesmo fim, chamado SWAP (*Simple Workflow Acess Protocol*) (SWENSON, 1998).

Os aspectos cobertos pela interface 4 são:

- Seleção
- Instanciação
- Encenação

de processo conhecidos, isto é, processos conhecidos *a priori* por ambas as partes envolvidas na colaboração. A máquina de *workflow* requisitante pode, opcionalmente, receber informações relativas à encenação do processo solicitado, depois de sua conclusão. As operações efetuadas entre as máquinas, tanto a requisitante quanto a requisitada, são passíveis de serem registradas para posterior auditoria.

➢ Modelos de Interoperabilidade

A WfMC identificou quatro modelos de interoperabilidade, descritos a seguir.

1. Processos Encadeados

Neste modelo de interoperabilidade, a instância de processo sendo encenada numa máquina de *workflow* A dispara a criação e encenação de uma instância de subprocesso numa máquina de *workflow* B. Uma vez que a encenação do subprocesso em B tenha começado na máquina B, a máquina A pode terminar ou continuar com a encenação de sua instância de processo, sem se interessar mais pela instância criada (Figura 2.7).

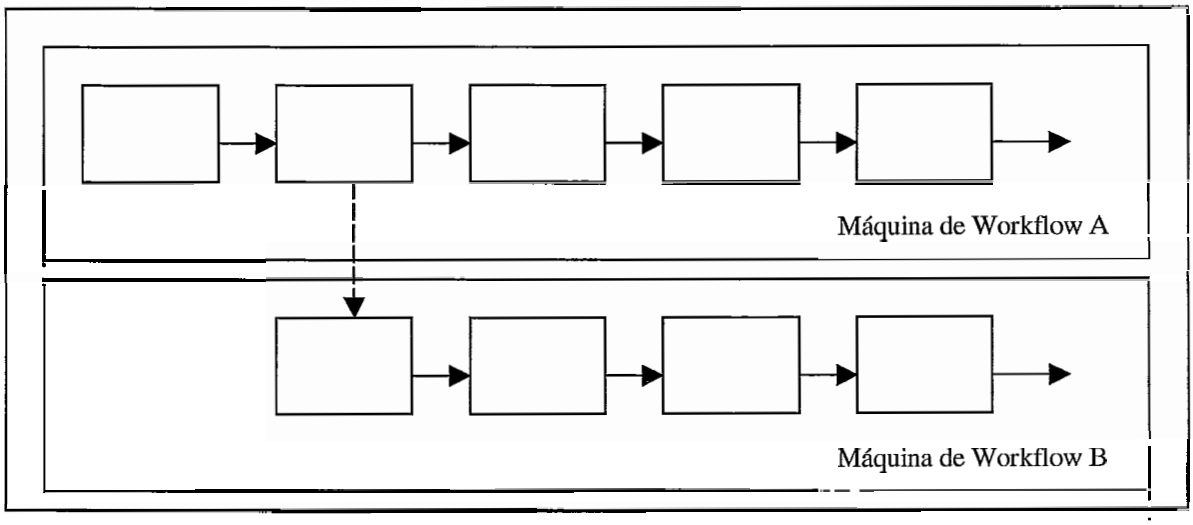

Figura 2.7 – O Modelo de Interoperabilidade Encadeado (WfMC-TC-1012, 1999)

2. Subprocesso Síncrono Encadeado

Esse modelo assume que uma instância de processo encenada numa máquina de *workflow* causa a criação e encenação de uma instância de subprocesso numa segunda máquina de *workflow*. A atividade do *workflow* requisitante permanece ativa até que o subprocesso termine de algum modo, quando então o controle é retornado à atividade que o invocou. A sincronização é obtida através de notificações nos valores de atributos da instância do processo ou no estado da instância do subprocesso (Figura 2.8).

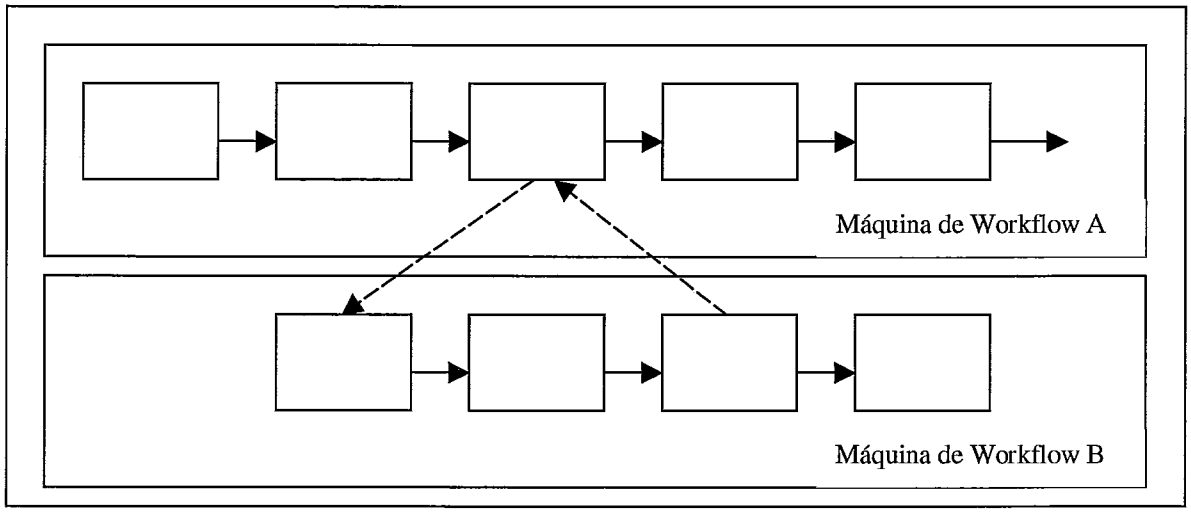

Figura 2.8 – O Modelo de Interoperabilidade Subprocesso Encadeado (WfMC-TC-1012, 1999)

3. Subprocesso Sincronizado por Evento

Esse modelo permite que eventos sejam disparados em um processo sendo encenado numa máquina de *workflow* diferente. Esse disparo de evento pode ocorrer devido ao subprocesso sendo abortado pela sua máquina de *workflow* ou como parte de um *check-point* lógico definido entre duas instâncias de processo sendo encenadas em máquinas de *workflow* diferentes (Figura 2.9).

4. Subprocesso Encadeado (Síncrono)

A Figura 2.10 apresenta o modelo Subprocesso Encadeado (Síncrono), que permite que uma máquina de *workflow* crie uma instância de processo em outra máquina e provoque o início de sua encenação. A máquina de *workflow* requisitante continua a encenação da instância de processo que invocou o subprocesso até que aquela atinja um ponto onde ela fará uma sincronização com o subprocesso. Nesse momento, ela sonda o

subprocesso para verificar se esse já está completo. Caso o subprocesso ainda esteja em andamento, a máquina requisitante espera o subprocesso terminar antes de dar prosseguimento à encenação do processo pai. É possível que o subprocesso tenha sido abortado antes do processo pai atingir o ponto de sincronização. Nesse caso, a máquina requisitante pode colher informações sobre a causa da interrupção do subprocesso e dar prosseguimento à encenação do processo pai.

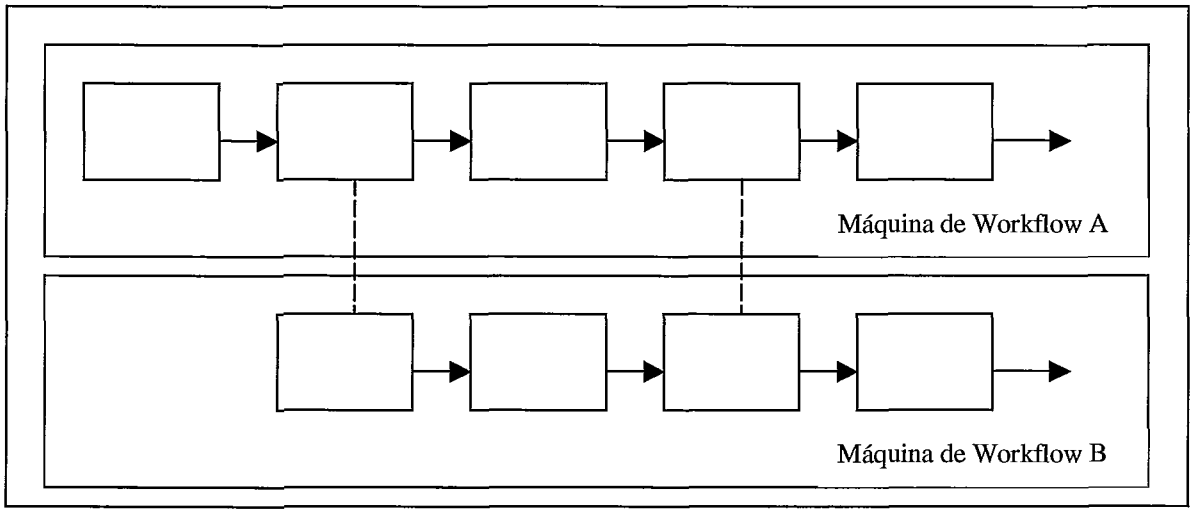

Figura 2.9 – Sincronização de Eventos (WfMC-TC-1012, 1999)

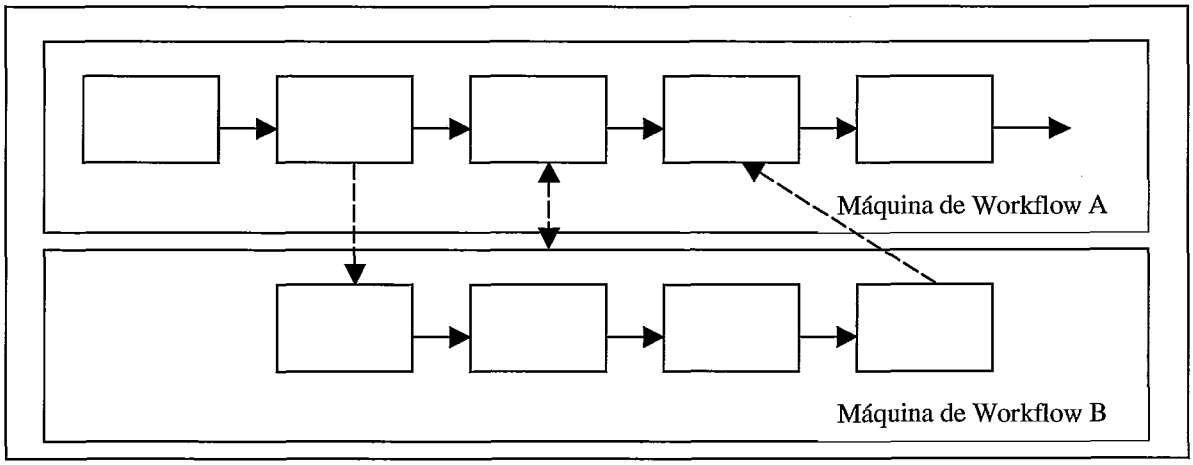

Figura 2.10 – O Modelo de Interoperabilidade Subprocesso Encadeado (Síncrono) (WfMC-TC-1012, 1999)

➢ **Operações**

A especificação define as seguintes operações, visando a comunicação entre as máquinas de *workflow*:

- **Alterar estado de instância de processo**

  Instrução para alterar o estado de uma determinada instância de processo que está sendo encenada em outra máquina de *workflow* do seu estado atual para o estado designado.

- **Criar instância de processo**

  Instrução para uma máquina de *workflow* criar uma nova instância de processo baseada na definição de processo designada.

- **Obter atributos de instância de processo**

  Instrução para recuperar os valores dos atributos de instância de processo requisitados (dados relevantes de processo).

- **Obter estado de instância de processo**

  Instrução para obter o estado de uma instância de processo que está sendo encenada em outra máquina de *workflow.*

- **Disparar atividade**

  Instrução para outra máquina de *workflow* disparar uma atividade designada de uma instância de processo que está sendo encenada

- **Listar instâncias de processo**

  Retorna uma lista de PIDs (identificações de instâncias de processos) selecionadas pelo critério dado no filtro passado como parâmetro.

- **Atributos de instância de processo alterados**

  Instrução para outra máquina de *workflow* para ajustar os valores de uma determinada instância de processo (dados relevantes de processo) para uma dada instância de processo.

- **Estado de instância de processo alterado**

Notificação para outra máquina de *workflow* sobre a mudança de estado em um sub(processo) no qual ela registrou interesse.

- **Abandonar instância de processo**

  Notificação para outra máquina de *workflow* indicando que ela pode liberar toda memória usada para armazenar estruturas de dados pertencentes à instância de processo determinada e/ou não se incomodar em enviar mensagens de notificação referentes à encenação dessa instância de processo.

- **Estabelecer atributos de instância de processo**

  Instrução para outra máquina de *workflow* que estabelece os valores de atributos de instância de processo (dados relevantes de processo) para uma dada instância de processo que está sendo encenada.

Contratos

A interface 4 assume que as empresas que queiram realizar a interoperabilidade de processos o façam no contexto de um contrato de interoperabilidade. O contrato é estabelecido de acordo com a natureza e os requisitos dos processos sendo encenados entre as organizações e os requisitos das próprias organizações. Os itens que constam de um contrato de interoperabilidade são:

1. Que máquinas de *workflow* dentro de um domínio de serviços são capazes de operar com máquinas de *workflow* do outro domínio;
2. Que definições de *workflow* podem ser encenadas dentro de um domínio de serviços ao comando de máquinas de *workflow* no outro domínio de serviços;
3. O protocolo de transporte suportado (por ex., MIME, Wf-XML, CORBA, etc.)
4. Para cada definição de *workflow* presente no contrato:
   - O perfil de conformidade necessário para efetuar a interoperabilidade
   - Requisitos de entrada/instanciação
   - Para cada item compartilhado de dado relevante do *workflow*:
     - Direitos de acesso (leitura/escrita)

        - Restrições de valores
    - Saídas/Resultados/Elementos de dados relevantes de *workflow* retornados
    - Política de auditoria de dados:
        - Operações a serem registrados
        - Atributos a serem registrados
    - Política de controle de alterações
5. Política de segurança e implementação
    - Autenticação
        - Máquinas de *workflow*
        - Papéis e identidades de usuários
        - Política sobre não repúdio
        - Criptografia e gerenciamento de chaves
        - Tratamento de brechas de segurança
6. Tratamento de exceções/Protocolos de recuperação e transações
7. Acordo sobre nível de serviço, mensuração e penalidades de desempenho

2.1.3. Ciclo de Vida de Instâncias de Processo e Atividade

O serviço de encenação de processos pode ser visto como uma máquina de estados, onde instâncias de processos e atividade mudam de estado em resposta a eventos externos ou a decisões de controle tomadas pela máquina de *workflow* (WfMC-TC-1003, 1995).

O esquema de transição de estados de uma instância de processos é mostrado na Figura 2.11.

Os estados que uma instância de processo podem assumir são:

- Iniciado: A instância de processo foi criada, mas ainda não está sendo executada.
- Executando: Foi iniciada a execução da instância de processo e uma ou mais das suas atividades podem ter sido iniciadas.
- Ativo: uma ou mais atividades foram iniciadas e existem instâncias de atividades.

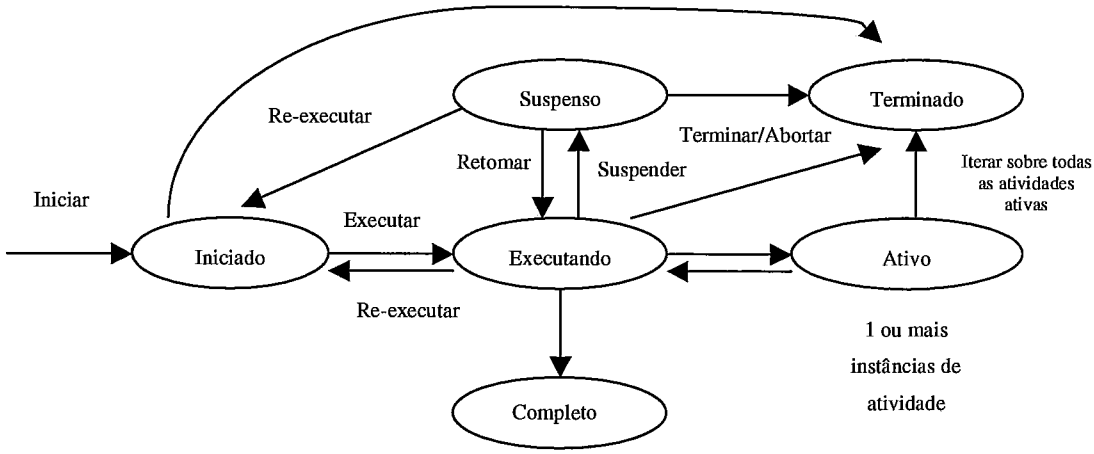

Figura 2.11 – Ciclo de Vida de uma Instância de Processo (WfMC-TC-1003, 1995)

- Suspenso: a instância de processo está suspensa e atividades não podem ser iniciadas até que a execução dessa instância de processo seja retomada.
- Completo: a instância de processo atingiu seu final, isto é, foi completada com sucesso.
- Terminado: a execução do processo foi parada (anormalmente), por erro ou comando do usuário.
- Arquivado: a instância de processo foi posta num estado de arquivamento. Pode ser recuperada posteriormente.

Na Figura 2.12 pode ser visto o esquema de transição de estados de uma instância de atividade.

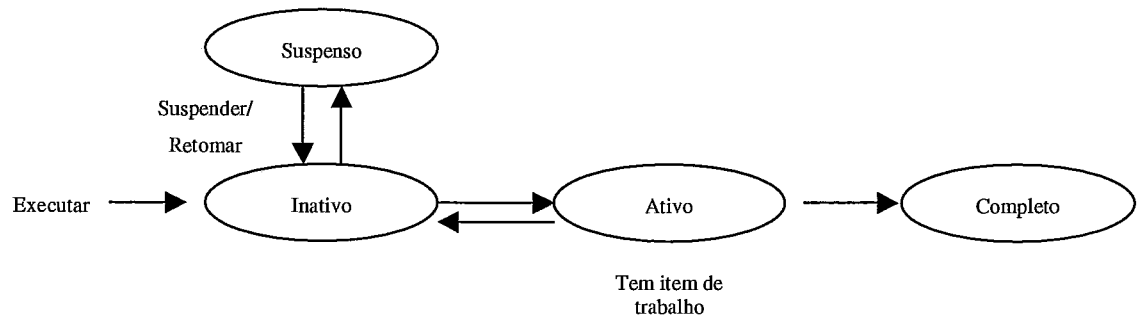

Figura 2.12 – Ciclo de Vida de uma Instância de Atividade (WfMC-TC-1003, 1995)

Os estados que uma instância de atividade podem assumir são:

- Inativo: a instância de atividade foi criada, mas não ativada. Ainda não existem itens de trabalho (*work itens*) para essa atividade.
- Ativo: um ou mais itens de trabalho (*work itens*) foram criados e assinalados para processamento.
- Suspenso: a execução da instância de atividade está suspensa. Nenhum item de trabalho pode ser iniciado até que essa instância seja retomada.
- Completo: a execução da instância de atividade chegou ao fim, isto é, foi finalizada com sucesso.

Um produto particular de *workflow* pode definir outros estados e transições. A WfMC não padroniza o comportamento interno de sistemas de *workflow*, mas apenas fornece os conceitos básicos para situar o contexto adequado dos efeitos dos comandos da API estabelecida.

2.2. OMG – *Workflow Management Facility*

O objetivo da *Workflow* Management Facility, criada pela OMG (Object Management Group), é prover um *framework* orientado a objetos que permita que produtos de *workflow* trabalhem conjuntamente, de maneira compatível (OMG-WMF, 2000). Esse *framework* foi especificado como um conjunto de interfaces, definidas na linguagem IDL. O trabalho da OMG está baseado e em conformidade com o trabalho da WfMC.

2.2.1. Modelo Conceitual da *OMG Workflow Management Facility*

A Figura 2.13 mostra o modelo conceitual de dados estabelecido pela OMG para sistemas de *workflow*, onde podem ser vistas as interfaces componentes do *framework* para sistemas de *workflow*. A seguir será feita uma breve descrição de cada uma das classes presentes no modelo da OMG:

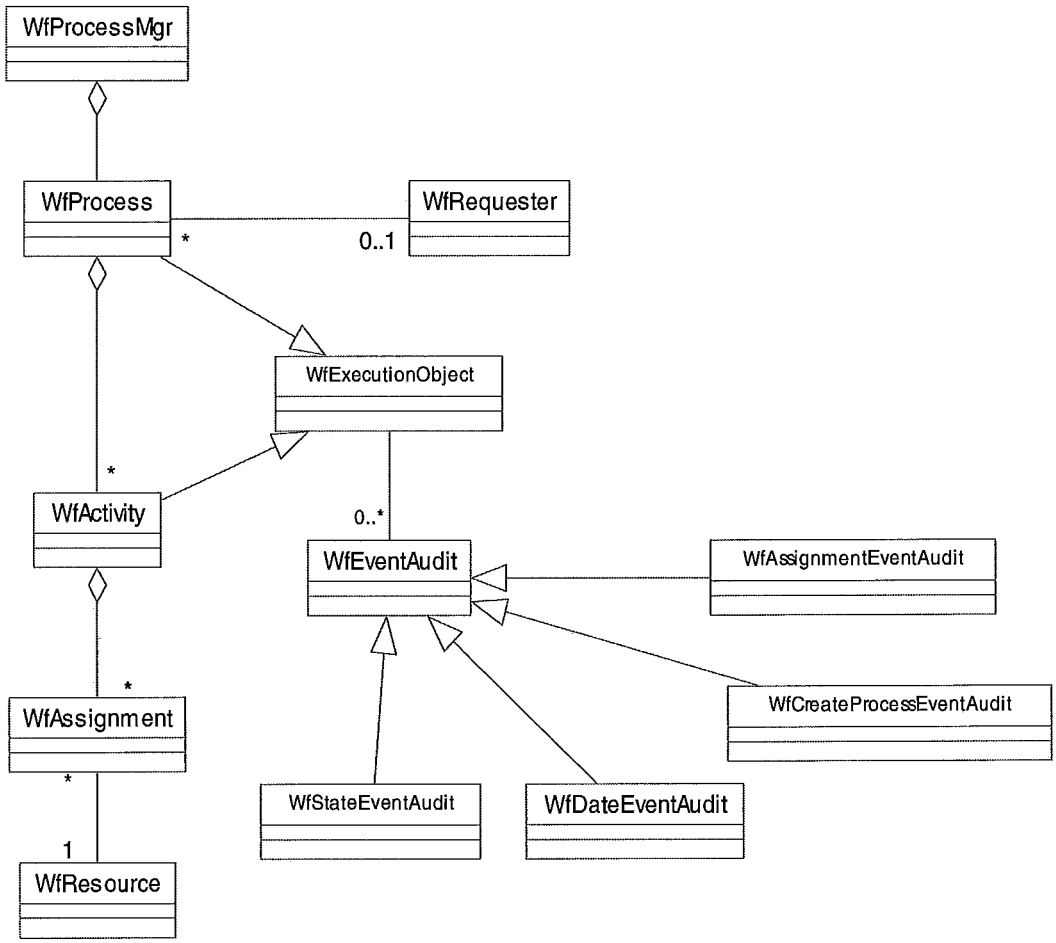

Figura 2.13 – Modelo Conceitual da OMG *Workflow Management Facility*

WfRequester: Liga o dono imediato de um pedido de processo. Esse objeto receberá eventos relativos ao processo, como, por exemplo, processo completado com sucesso.

WfProcessMgr: Classe auxiliar que funciona como "fábrica" de processos.

WfProcess: O executor de um pedido de *workflow* expedido por um ator automatizado.

WfActivity: Um passo em um WfProcess.

WfExecutionObject: Uma classe abstrata base para WfProcess e WfActivity.

WfAssignment: Relaciona atividades a recursos.

WfResource: Uma pessoa ou coisa que pode fazer e aceitar uma atividade.

WfEventAudit: Uma classe genérica para registrar eventos de *workflow*. Suas subclasses registram eventos mais específicos ocorridos no sistema de *workflow*.

2.2.2. Ciclo de Vida de Objetos de *Workflow*

A *Workflow Management Facility* define uma máquina de estados com estados aninhados, isto é, com estados e subestados (Figura 2.14). Esses estados se aplicam a "Objetos de Execução" (WfExecutionObject), classe genérica da qual descendem tanto processos quanto atividades (WfProcess e WfActivity).

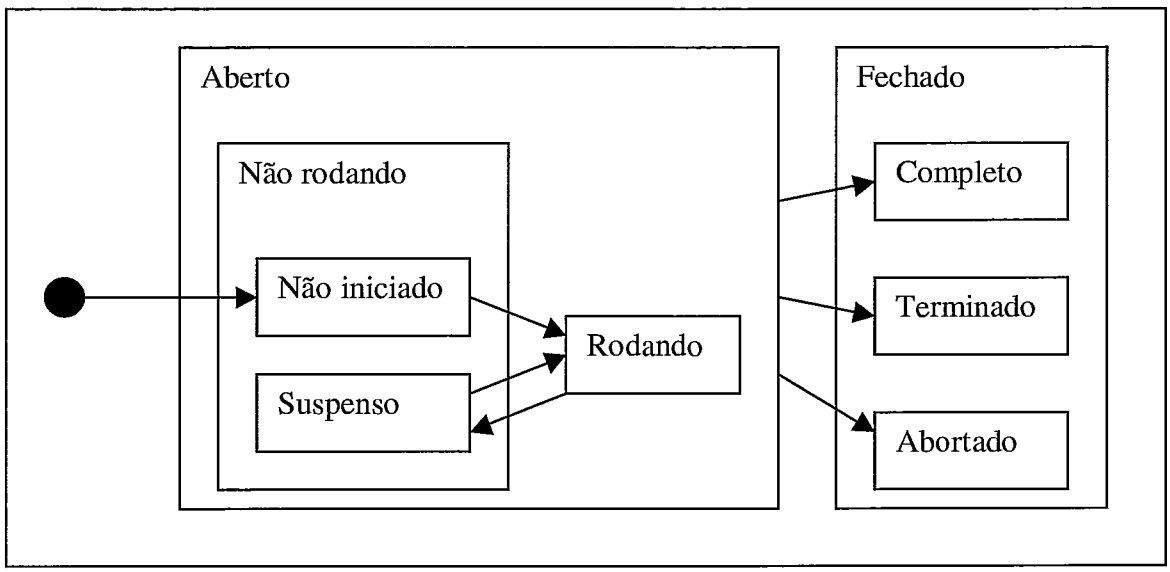

Figura 2.14 – Estados de um WfExecutionObject (OMG-WMF, 2000)

Os estados que um objeto de execução (WfExecutionObject) podem assumir são:

- Aberto: o objeto está ativo.
  - Não rodando: o objeto está ativo e pronto para ser executado.
    - Não iniciado: provê um estado após sua criação que indica que o objeto está ativo e pronto para ser iniciado.
    - Suspenso: a execução do objeto está temporariamente parada.
  - Rodando: o objeto está ativo e em execução
- Fechado: o objeto está finalizado e inativo.
  - Completo: a execução do objeto finalizou sua tarefa, quer dizer, foi completada com sucesso. Assume-se que todos os objetos dependentes deste também estão completos.
  - Terminado: A execução do objeto foi parada antes do seu fim normal. Assume-se que todos os objetos de execução dependentes deste objeto estão completos ou terminados quando o objeto entra nesse estado.

- Abortado: A execução do objeto foi abortada antes do seu fim normal, mas nada se pode assumir a respeito dos estados dos objetos dependentes destes.

2.3. ebXML

O padrão ebXML (EBXML, 2001) é um conjunto modular de especificações técnicas baseadas em XML (W3C-XML, 2000), que permite que empresas de qualquer tamanho ou localização possam conduzir negócios na Internet, formando assim um mercado global.

Uma característica fundamental do ebXML, no sentido de interoperabilidade, é ir além da simples troca de documentos eletrônicos, como acontece no caso da tecnologia de EDI (EDIFACT, 2002) e adotar uma visão de negócios baseada em transações e orientada a processos. Aproveita a vasta experiência de EDI e tem como requisito permitir uma migração suave dos sistemas que utilizam EDI para esse novo *framework* padrão.

Os principais componentes do *framework* ebXML são: *Collaboration Protocol Profile (CPP), Collaboration Protocol Agreement (CPA), Business Process and Information Modeling, Core components, Messaging Service* e *Registry/Repository.*

➢ *Collaboratin Protocol Profile* (CPP)

Um CPP descreve as ofertas de uma empresa numa forma padrão. Descreve a capacidade de troca de mensagens e processos de negócios que uma empresa oferece. A descrição é bilateral, isto é, é feita tanto para a empresa no papel de contratante de serviços como no papel de contratada.

➢ *Collaboratin Protocol Agreement* (CPA)

Um CPA descreve o acordo feito entre duas empresas para conduzir um determinado negócio em conjunto. Pode ser gerado automaticamente (ou semi-automaticamente) a partir de dois CPPs.

➢ *Business Process and Information Modeling*

O *framework* ebXML oferece um meta-modelo que permite a especificação de

processos. Um processo é constituído de transações, fluxos de documentos, colaborações bilaterais, formatos de dados, etc. Os seguintes conceitos são oferecidos por esse meta-modelo:

Colaborações de negócios: é um conjunto de transações de negócios entre parceiros. Cada parceiro desempenha um ou mais papéis na colaboração. Uma colaboração é expressa como um conjunto de atividades realizadas entre papéis, sendo que uma atividade pode ser uma transação ou uma outra colaboração, permitindo assim a utilização de recursividade na definição de colaborações.

Transações de negócios: é o conceito central na interação entre duas entidades de negócios. Uma transação é conduzida entre dois parceiros desempenhando papéis opostos. O resultado de uma transação é sucesso ou falha. Transações representam a unidade atômica de trabalho, o que significa que após uma transação deve ser possível executar uma ação de *rollback* ou *commit*.

Fluxo de documentos: uma transação é realizada como um fluxo de documentos entre duas entidades de negócios, uma com papel de requisitante e outra de requisitada. Há sempre um documento de requisição. Opcionalmente, pode haver um documento de resposta. Há fluxos que não exigem resposta, como no caso de notificações.

Coreografia: uma coreografia descreve o ordenamento e transições entre as transações de negócios ou entre subcolaborações dentro de uma colaboração. Corresponde ao diagrama de atividades da UML.

- *Core Components*

Core Components são formatos de documentos padrões de negócios, tais como contratos, ordens de compra, faturas, etc. Grande parte destes documentos originou-se de padrões de troca de documentos desenvolvidos anteriormente, como EDIFACT (EDIFACT, 2002) e RosettaNet (ROSETTANET, 2002). Esses documentos são gerais para quaisquer ramos de negócios, não estando alinhados a nenhum setor vertical em específico.

➢ *Messaging Service*

O ebXML define um protocolo que permite a troca de mensagens entre empresas com toda a semântica presente em sistemas *MOM (Message-Oriented Middleware)*, como troca assíncrona ou síncrona e opções de confiança (*reliability*). É construído sobre o protocolo SOAP (W3C-SOAP, 2000), com extensões para anexo de documentos, segurança, entrega confiável de mensagens e tratamento de erros. É independente de protocolo de comunicação de transporte: pode ser utilizado sobre http, ftp, smtp, etc.

➢ *Registry/Repository*

Um dos componentes do *framework* ebXML é um registro/repositório que armazena *CPPs, CPAs, ebXML Core Components*, entre outros documentos quaisquer. O Registro permite que usuários consultem a base em busca de componentes relevantes e potenciais parceiros de negócios. Os serviços armazenados em um ebXML *Registry* podem ser publicados via UDDI (UDDI, 2000), que é um mecanismo para publicação de serviços. Enquanto o UDDI fornece somente referências a serviços, o ebXML *Registry/Repository* disponibiliza os serviços em si e os documentos associados a eles.

2.4. Conclusões Preliminares

Este capítulo apresentou dois dos mais importantes padrões no domínio de *workflow*, feitos por duas entidades de grande crédito: WfMC e OMG. Mais especificamente, foram mostradas duas interfaces definidas pela WfMC que terão grande importância para esse trabalho (interfaces 2 e 4). Estudou-se ainda como são definidos os ciclos de vida de instâncias de processos e atividades segundos essas duas entidades. Esse estudo servirá de base para a definição dos estados a serem monitorados pelo ambiente, como será visto mais adiante, no Capítulo 5.

Foi apresentado também um padrão de comércio eletrônico entre empresas, chamado ebXML, que apresenta alguns conceitos que serão utilizados na infra-estrutura proposta neste trabalho.

3 – Problemas relacionados à Integração de Processos entre Empresas

Nesse Capítulo, serão discutidos os principais problemas relacionados à questão de integração de processos entre empresas. De um modo geral, esses problemas têm como origem o seguinte dilema: permitir que uma empresa utilize, monitore satisfatoriamente e controle os serviços fornecidos pelos processos de outra empresa, porém sem interferir na privacidade e autonomia desses processos.

Entre as abordagens adotadas atualmente para a integração de processos, privilegia-se o primeiro aspecto em detrimento do segundo, ou seja, permite-se algum monitoramento e controle sobre os processos, mas fere-se sua privacidade e autonomia, como é o caso da especificação da interface 4 da WfMC. Algumas soluções foram propostas para tratar essa questão, embora nenhuma seja satisfatória no tratamento de um conjunto mínimo dos problemas. Isto é, as soluções propostas resolvem somente parte dos problemas.

Na primeira seção desse Capítulo (Seção 3.1), cada um dos problemas será detalhado e discutido. Essa análise constitui uma base para a especificação de requisitos do ambiente proposto por esse trabalho, apresentada no Capítulo seguinte. Na Seção 3.2 serão apresentados alguns sistemas que visam minimizar ou resolver os problemas discutidos.

3.1. Alguns Problemas relacionados à Integração de Processos entre Empresas

A lista de problemas discutidos nesta Seção foi elaborada levando-se em consideração sua relevância para o domínio de integração de empresas, segundo a discussão presente em trabalhos desenvolvidos nessa área e em relatórios sobre resultados de conferências e *workshops* realizados especificamente sobre o assunto (GEORGAKOPOULOS et al., 1999a, GREFEN et al., 2000, LUDWIG e HOFFNER, 1999, KLINGEMANN, 2000, KLINGEMANN et al., 1999, LUDWIG et al., 1999, ALONSO et al., 1999, LAZCANO et al., 2001, LAZCANO e ALONSO, 2001, CASATI et al., 2001, etc.). Os principais aspectos cobertos são: heterogeneidade de ambientes,

autonomia de negócios, escolha dinâmica de serviços e controle e monitoramento sobre serviços prestados.

3.1.1. Sistemas Heterogêneos de Encenação de Processos

Um dos problemas básicos encontrado na integração de processos entre diferentes empresas diz respeito às diferenças técnicas existentes entre os sistemas em que esses processos são encenados.

Uma face deste problema refere-se à linguagem utilizada para modelagem e definição dos processos. Todo sistema de gerenciamento de workflow possui uma linguagem de definição de processos que é utilizada para permitir que os processos modelados nas ferramentas de definição de processos sejam incluídos na máquina de workflow e, algumas vezes, para o próprio armazenamento dessas definições internamente. No entanto, cada sistema pode utilizar uma linguagem própria, proprietária. Isso impede que outras empresas "entendam" esses processos, a fim de incorporá-los às suas próprias definições.

Outro aspecto técnico de divergência entre máquinas de workflow é a forma de invocação de execução de processos que permita comunicação entre elas, como apontado em (LUDWIG et al., 1999). Por exemplo, diferentes primitivas de interação podem ser oferecidas, como "iniciar processo", "abortar processo", "suspender processo", etc. Portanto, uma divergência entre os conjuntos de mensagens e sua semântica disponibilizados pelos diferentes WfMSs pode dificultar a integração de processos.

Por último, pode haver diferenças quanto aos protocolos de comunicação utilizados como transporte das mensagens trocadas entre os sistemas de workflow. Alguns dos protocolos mais utilizados são: SMTP e MIME, XML sobre HTTP e CORBA IIOP.

3.1.2. Definições de Processos e Estruturas de Dados Heterogêneas

Empresas distintas, embora ofereçam o mesmo serviço, podem possuir diferentes maneiras de implementá-lo internamente, no nível de processo. Ou seja, cada empresa possui a sua maneira própria de realizá-lo, o que é refletido na estruturação de seu processo de fornecimento do serviço. O que ocasionam essas diferenças são o mercado alvo do

serviço (interesses de seus clientes), a capacidade e a estrutura interna da empresa, o grau de aprimoramento pela qual o processo já passou ou ainda aspectos estratégicos de competitividade, no intuito de oferecer um serviço diferenciado a seus clientes.

Essa questão pode ser ilustrada através de um exemplo baseado em outro exemplo dado por GEORGAKOPOULOS et al. (1999a). Um mesmo serviço de fornecimento de ligações locais é oferecido por três empresas de telefonia distintas, que o implementam cada qual com um processo diferente dos demais (Figura 3.1). O primeiro processo contém três macro-atividades: registro da solicitação de serviço, fornecimento técnico e cobrança do serviço, enquanto no segundo processo há somente duas macro-atividades, pré-solicitação e pós-solicitação, que juntas realizam as mesmas tarefas dispostas nas três macro-atividades do primeiro. Há ainda um terceiro processo onde a atividade de cobrança vem antes de qualquer atividade de trabalho técnico. Uma empresa que quisesse integrar os serviços dessas empresas ao seu processo, por exemplo, uma empresa de telefonia de longa distância, teria que prepará-lo para incorporar as três possibilidades de fornecimento do serviço.

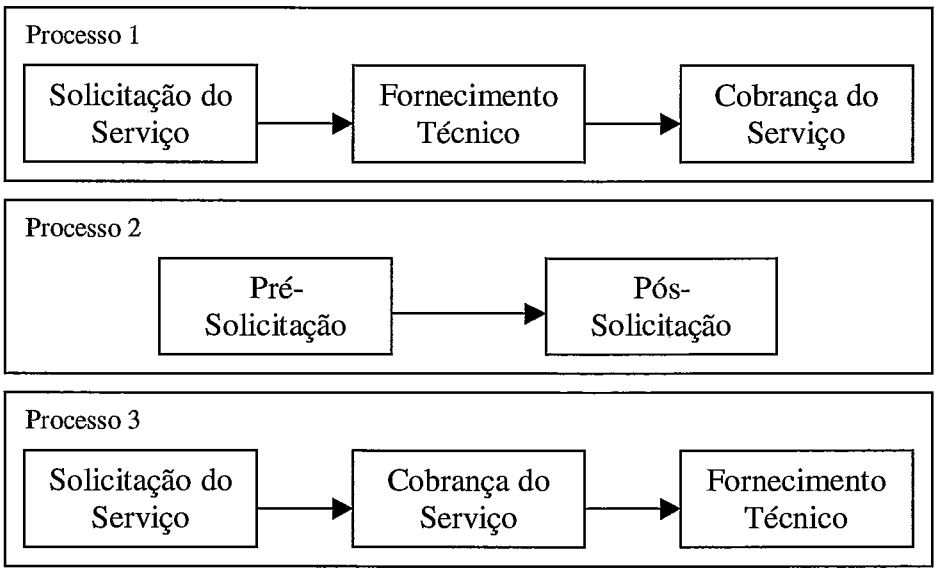

Figura 3.1 - Diferentes Implementações para um mesmo Serviço

Além do processo em si, há a questão da definição das estruturas de dados utilizadas pelos processos (dados relevantes do workflow). Cada uma das empresas do exemplo anterior poderia aceitar o pedido de fornecimento do serviço de um modo diferente. Na primeira, o pedido poderia conter somente os dados do cliente e a localidade da linha. Na

segunda, poderiam constar além destes dados também dados técnicos. Por último, a terceira poderia exigir dados de cobrança relativos ao cliente como entrada do serviço, antes de sua realização. É preciso levar em consideração ainda a estrutura interna dos documentos trocados. Por exemplo, os dados que definem um registro de cliente podem estar organizados de forma diferente ou com diferente semântica entre as fornecedoras do serviço. Logo, a empresa que contratasse esses fornecedores de serviços teria que realizar três mapeamentos diferentes de seus dados para os dados do serviço contratado, a fim de utilizar esses serviços.

3.1.3. Ausência de Privacidade de Processos

Integração de empresas no nível de atividades de processos implica que as empresas compartilhem definições de processos, pois para uma empresa contratar uma fornecedora de serviço ela deve conhecer os detalhes do processo contratado, incluindo as atividades que o compõe. Esse problema foi discutido anteriormente quanto a aspectos técnicos: as empresas têm que compreender de alguma forma a linguagem de definição de processos umas das outras. Porém, além do aspecto técnico, ainda mais crucial é o problema da privacidade do processo, quando se fala deste tipo de abordagem de integração.

Numa empresa orientada a processos, as formas como estes são definidos e executados constituem informações valiosas que devem ser mantidas secretas, pois é através dos seus processos que a empresa consegue se diferenciar estrategicamente no mercado. Ao observar-se, medir-se e analisar-se um processo, podem-se descobrir importantes detalhes sobre a eficiência de uma empresa para entregar um serviço ou produto, como, por exemplo, custo, tempo, número de funcionários e recursos empregados no processo. Por outro lado, através destes mesmos números, fraquezas inerentes à empresa também podem ser reveladas (GEORGAKOPOULOS et al., 1999a).

Portanto, é altamente compreensível que as empresas não queiram expor seus processos a seus parceiros, clientes e colaboradores, muito menos possibilitar que seus concorrentes os espiem, a fim de realizarem a integração de processos.

3.1.4. Restrição de Independência e Autonomia de Processos

Uma estratégia utilizada pelas empresas e reconhecida como eficaz, além de indispensável nos dias atuais, na tentativa de se manterem competitivas é a melhoria contínua de processos, adotada a partir de uma visão de reengenharia de negócios (HAMMER e CHAMPY, 1993, DAVENPORT, 1993, JACOBSON, 1995). As empresas estão sempre procurando realizar seus processos de produção de maneira mais eficiente, ou seja, com menor custo e resultando em maior qualidade do produto final ou serviço oferecido.

Um dos pontos negativos das tecnologias atuais disponíveis para a integração de empresas é obrigá-las a vincular fortemente seus processos internos aos de suas parceiras, tornando-os muito dependentes deles. Todos os processos internos devem ser adaptados para comportar a integração, o que os faz deixarem de existir como entidades autocontidas e independentes.

É desejável também que os processos permaneçam autônomos em relação a quem os contrata. Se a forma de contratação de um serviço depender da estrutura interna de seus processos, como acontece na maioria dos sistemas atuais, também haverá dificuldades em relação à evolução de processos. Existem duas facetas desse mesmo problema: uma se refere aos contratos de interação em vigência, que precisariam ser alterados para as contratações que viriam a acontecer. Ainda mais grave é a questão dos processos que estão em andamento, pois isso obriga a manutenção e encenação de diversas versões de definições de um mesmo processo por uma máquina de workflow. Algumas soluções foram propostas para a questão de adaptação dinâmica de processos e tratamento de mudanças em execução (CASATI, 1998, HAN et al., 1998), porém para processos executados no contexto interno de uma empresa. No âmbito de processos que extrapolam os domínios de uma só empresa, esse problema ainda é pouco explorado (GREFEN et al., 2000).

A alta dependência e a perda de autonomia impedem ou dificultam que a empresa tenha liberdade para alterar e evoluir seus processos. Isso a deixa pouco flexível quanto a mudanças no mercado e acarreta um alto custo associado às mudanças imprescindíveis feitas nos processos (o que implica em redefinir todas as interações com processos externos), e, assim, suscetível à perda de competitividade estratégica de mercado.

3.1.5. Ausência de Meios para se Escolher Dinamicamente Serviços

É reconhecido que um mesmo serviço pode ser oferecido por diversas empresas concorrentes. Cada empresa o executará a sua maneira e o resultado para o contratante também terá diferenças quanto a preço, qualidade e tempo, entre outros fatores.

Uma empresa, ao procurar os serviços de uma outra, fará a seleção baseando-se na melhor relação custo/benefício para um determinado contexto, num determinado momento, de modo a atingir seus objetivos de negócios. Por exemplo, poder-se-ia dizer que o critério principal de seleção de serviços de uma dada empresa é a qualidade do serviço. Porém, em determinadas épocas do ano onde o aumento da procura pelos seus produtos é maior (no Natal, por ex.), o critério tempo passa a possuir um peso maior, respeitando-se um critério de qualidade mínima.

Logo, pode-se perceber que a escolha de um serviço é dinâmica, ou seja, é determinada de acordo com o momento em que o processo é executado e não *a priori*, quando o processo é definido. No entanto, as primitivas oferecidas pelos meta-modelos tradicionais de workflow (por ex., WFMC-TC-1016, 1999, OMG-WMF, 2000) não suportam a escolha dinâmica de serviços. Segundo esses meta-modelos, para uma atividade ser executada durante a encenação do processo ela deve estar explicitamente declarada em sua definição.

3.1.6. Explosão de Especificação

Utilizando-se as linguagens tradicionais para modelagem de processos, em cada processo que interaja com processos de outras empresas devem ser adicionadas atividades de chamada do processo provedor e, provavelmente, também atividades de *feedback* para cada situação possível em que esse processo passe que seja de interesse da empresa cliente.

Além disso, a questão da escolha dinâmica do realizador de uma tarefa entre múltiplos provedores de serviços requer a modelagem de novas atividades que contemplem cada combinação possível de escolha desses provedores.

Esses dois fatores juntos levam a uma explosão combinatória de definições de processos para expressar-se num modelo a integração desejada, resultando num modelo de processo bastante complexo e por isso, difícil de ser coordenado, sujeito a falhas e de

manutenção onerosa. Definições de processos assim são praticamente impossíveis de serem compreendidas e tratadas por seres humanos, o que prejudica o trabalho de reengenharia e melhoria contínua de processos e compromete a sobrevivência da empresa. A Figura 3.2 apresenta um exemplo de definição gráfica de processo com múltiplos provedores de serviços e atividades extras para chamada e retorno de serviços.

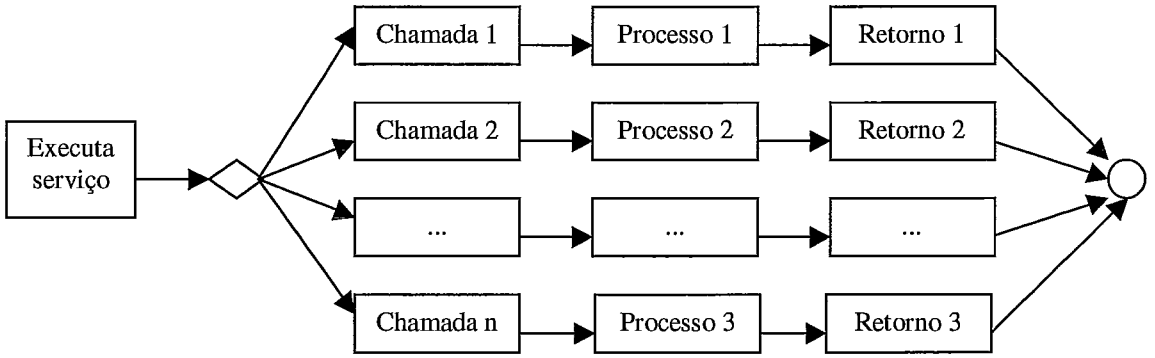

Figura 3.2 – Explosão de Especificação

3.1.7. Descoberta e Seleção de Serviços

A necessidade de escolha dinâmica de serviços remete a duas outras questões: como encontrar os serviços disponíveis que possam atender à necessidade da empresa e, dentre eles, como selecionar o serviço mais adequado, em tempo de execução.

A primeira questão possui diversos aspectos: os serviços precisam ser descritos de forma que quem os encontre entenda sua utilidade, conheça seus critérios de qualidade de serviço, além de fornecer informações técnicas a respeito de sua contratação e execução.

Para que um serviço seja descoberto, ele deve estar em algum local público, que possa ser consultado pelas organizações interessadas em contratá-lo. Diversas arquiteturas de repositórios de serviços ou de localização de serviços estão sendo criadas recentemente para esse fim. CORBA Trading Service (OMG-TS, 1996), UDDI (UDDI, 2000) e ebXML Registry (EBXML-TA, 2001) são exemplos de tais mecanismos. Ainda assim, ainda não se chegou a um formato satisfatório de descrição de serviços, o que dificulta a sua descoberta.

Uma vez descobertos os serviços que possam vir a serem utilizados pela empresa, é preciso escolher dentre eles o mais adequado para o contexto atual. Para isso, precisam ser desenvolvidos algoritmos, heurísticas e mecanismos inteligentes que consigam, a partir da

análise da descrição do serviço, inferir a melhor escolha. Dados históricos, como média de tempo de execução, custo associado, entre outros, também podem e devem ser computados na escolha do melhor serviço.

3.1.8. Recursos Limitados de Monitoramento e Controle de Processos

Nos modelos tradicionais de workflow, a utilização de processos de outras empresas é feita através da relação de processo e subprocesso. No entanto, esta abordagem é bastante limitada quanto às possibilidades de monitoramento. A coordenação entre processo e subprocesso se dá da seguinte maneira: quando o fluxo de controle chega na atividade executada pelo subprocesso, é passado para o subprocesso, que é executado inteiramente antes de fornecer quaisquer resultados de interesse do processo principal. Portanto, nesse modelo os subprocessos são semelhantes a caixas-pretas, sem oferecer grandes possibilidades para se saber o que ocorre em seus interiores.

A única informação disponível sobre um subprocesso invocado por outro processo é o estado de execução em que se encontra. No entanto, os meta-modelos de workflow utilizados pela maioria dos sistemas de gerenciamento de workflow disponibilizam somente um conjunto de estados genéricos, idêntico ao definido pela WfMC ou com pequenas variações. Os estados definidos pela WfMC são: inativo, "rodando", suspenso, terminado (com falha ou interrupção do usuário) e completo (finalizado com sucesso) (WfMC-TC-1003, 1995).

Essas informações, apesar de oferecerem alguma indicação sobre o ciclo de vida do processo, pouco acrescentam em relação ao andamento das atividades específicas da aplicação modelada. Por exemplo, ao monitorar o primeiro processo ilustrado na Seção 3.1.2, seria interessante saber se o processo está na fase de solicitação, fornecimento do serviço técnico ou cobrança do serviço. Entretanto, a não ser que se examinassem as atividades em si (com todos os problemas decorrentes disso, discutidos anteriormente), tudo que se poderia saber sobre esse processo é que ele está "em execução", sem saber em que fase de execução.

Quanto ao controle que pode ser exercido sobre os processos contratados, em geral, as opções também se restringem ao ciclo de vida genérico de processo. Os comandos de

controle são, então, iniciar, suspender, retomar, abortar e forçar finalização. Não é possível, por exemplo, instruir um processo a transferir o controle de uma atividade para outra.

3.1.9. Sincronização de Processos

De forma a realizar uma ação integrada e coordenada entre as atividades que compõem um processo virtual, essas atividades precisam ser sincronizadas umas com as outras. Muitas vezes, o início de um processo depende dos resultados obtidos em outros processos.

Embora o problema de sincronização de processos aconteça também em sistemas de gerenciamento de workflow que executam somente processos internos, ele é ainda mais complexo no caso de sincronização de processos interorganizacionais, pois, em geral, só se tem acesso limitado às estruturas internas dos processos externos. É difícil, por exemplo, controlar a dependência que acontece entre uma atividade interna de um processo em relação a atividades de outro processo em outra empresa.

Como exemplo, podemos pensar em três processos que precisam ser executados coordenadamente: o Processo A (principal) oferta um bem a clientes, da venda até a entrega, o Processo B produz esse bem e o Processo C o entrega efetivamente ao cliente. Portanto, o Processo A só ponde invocar o Processo C após o processo B ter terminado sua execução.

3.1.10. Controle de Transações Distribuídas

Processos executados distribuidamente, entre diversas organizações, estão sujeitos a falhas técnicas e exceções, embora muitas vezes precisem ser executados atomicamente. Para mantê-los consistentes, é preciso que sua execução esteja dentro de uma transação, semelhante ao conceito análogo encontrado em sistemas gerenciadores de bancos de dados. Após uma falha, deve ser possível realizar uma operação de *rollback* e traze-lo de volta a um estado consistente.

O problema é que esse tipo de processo é executado em diversas máquinas de workflow heterogêneas, sob diferentes administrações. Portanto, o mecanismo de transação

deve ser global, no sentido de extrapolar um único ambiente e ter permissões para realizar operações pertinentes em todas as máquinas onde o processo é encenado.

3.2. Análise de Alguns Sistemas para Integração de Processos

Nesta seção, serão analisados alguns sistemas que se propõem a tentar resolver a questão de integração de processos entre empresas. São projetos acadêmicos, que procuram estender a tecnologia de *workflow* para apoiar a encenação de processos virtuais.

3.2.1. WISE

O projeto WISE (LAZCANO et al., 2001, LAZCANO et al., 2000, ALONSO et al., 1999, LAZCANO e ALONSO, 2001) tem como objetivo prover uma infra-estrutura para apoiar o comércio eletrônico (*Business to Business*). Baseia-se num modelo que contém os seguintes conceitos: comunidades de negociação, processos de negócios virtuais e empresas virtuais.

Sua arquitetura é composta de quatro itens: definição de processos, encenação de processos, análise e monitoramento e coordenação e comunicação.

No WISE, processos virtuais são construídos utilizando-se os serviços fornecidos por diferentes empresas. O WISE provê um mecanismo de publicação de serviços e permite a definição de processos baseados nesses serviços. São utilizados um catálogo de serviços e uma ferramenta (comercial) de modelagem de processos, que funcionam de maneira integrada.

A encenação de processos virtuais é executada no motor WISE, que é baseado num trabalho anterior, chamado OPERA (ALONSO et al., 1997). Esse motor funciona de modo distribuído. Sua arquitetura interna está dividida em três camadas: serviços de banco de dados, que provêem funcionalidades de armazenamento e consulta de dados, serviços de processo, que permite a coordenação e monitoramento de processos; e serviços de interface, que fornecem meios de acesso ao sistema, por pessoas e aplicativos.

A infra-estrutura WISE fornece ferramentas para monitoramento de processos virtuais. Essa ferramenta é utilizada tanto para acompanhamento de execuções de instâncias de processo quanto para avaliar dados históricos de execuções passadas.

O projeto prevê a necessidade de comunicação entre os participantes envolvidos na execução de um processo virtual. Para isso, fornece uma ferramenta de comunicação baseada em contexto. Por exemplo, pode-se determinar que se quer comunicar com a pessoa que realizou a tarefa anterior no fluxo de trabalho, sem ter que indicar precisamente quem é essa pessoa.

3.2.2. CrossFlow

O projeto CrossFlow (GREFEN et al., 2000, LUDWIG e HOFFNER, 1999, KLINGEMANN, 1999) tem como objetivos prover meios para ligar WfMSs de duas organizações na descoberta de parceiros de negócios, casar suas ofertas e requerimentos, configurar seus sistemas e realizar o *outsource* de uma organização para outra, de acordo com o contrato estabelecido.

O enfoque do CrossFlow está baseado em quatro aspectos: *outsourcing* dinâmico de serviços, especificação de serviço baseada em contrato, interação avançada fina e configuração automática de infra-estrutura a partir de contrato.

Na abordagem CrossFlow, uma organização (requisitante) passa parte do seu serviço para outra organização (provedora). A escolha da organização provedora do serviço é feita dinamicamente, durante a execução do processo da organização requisitante.

A interação entre parceiros é completamente especificada em contratos. O contrato contém detalhes da provisão do serviço, como uma especificação do processo da organização provedora.

O sistema provê noções de interação fina, operacionalizadas em serviços de apoio à cooperação (CSS). Os CSSs permitem o controle e monitoramento remoto de processos, gerenciamento de transação interorganizacional, gerenciamento de segurança e autenticação, etc.

Quando um contrato é estabelecido entre um provedor e um consumidor, uma arquitetura de encenação de serviço é automaticamente configura, de maneira simétrica, em ambos os parceiros.

O ciclo de vida de uma relação de provisão/consumo de um serviço é dividido em quatro fases:

1) O consumidor procura uma oferta de serviço adequada e estabelece um contrato que especifica o serviço e aspectos relacionados com o provedor;
2) As duas organizações configuram seus sistemas;
3) O serviço é executado pela provedora. Interações podem ocorrer entre os dois processos;
4) O serviço termina e a infra-estrutura preparada é desfeita.

A Figura 3.3 ilustra o estabelecimento de um contrato na arquitetura CrossFlow. A arquitetura prevê o uso de sistemas comerciais de workflow (no caso, é usado o MQSeries Workflow da IBM) e sistemas de *back-end* (BES).

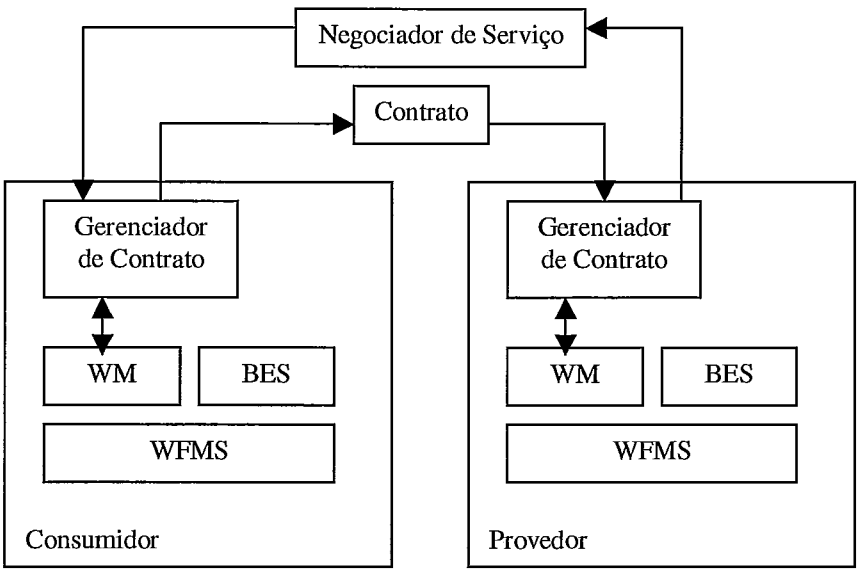

Figura 3.3 – Estabelecimento de um Contrato na Arquitetura CrossFlow

Uma característica interessante desse projeto é que ele fornece um modelo de transações distribuídas de processos, chamado X-transactional model (VONK et al., 1999). Entre as primitivas de controle oferecidas por esse modelo estão: parar, continuar, abortar, executar *rollback* e alterar variável. Esse modelo é implementado por um módulo de gerenciamento de transação na arquitetura.

3.2.3. CMI

O software CMI (Collaboration Management Infrastructure) (GEORGAKOPOULOS et al., 1999a, GEORGAKOPOULOS et al., 1999b, BAKER et al.,

1999) foi desenvolvido com o intuito de gerenciar processos de colaboração e prover percepção combinada de processos e situações. Ele foi concebido para tentar apoiar aplicações avançadas (por exemplo, resolução de crises e empresas virtuais) que não são suportadas adequadamente por sistemas convencionais de *workflow*.

Para atender a esses requisitos, o CMI provê um meta-modelo de processo – Collaboration Management Model (CMM) e um sistema orientado a componentes que o implementa. O CMM oferece primitivas para colaboração, baseando-se nos modelos de *groupware* e *workflow*.

O conceito de empresa virtual adotado pelo CMI consiste de várias empresas que se reúnem para oferecer um processo multiorganizacional (Multi Enterprise Process – MEP), que por sua vez é composto pelos processos de cada organização (Single Enterprise Process – SEP).

➢ CMM (Collaboration Management Model)

A infra-estrutura CMI está baseada num meta-modelo orientado a processo chamado CMM. Ele consiste de um modelo núcleo e algumas extensões, como pode ser observado na Figura 3.4.

O modelo núcleo (CORE) provê um conjunto de primitivas de processo que constituem a base para todas as outras extensões. As extensões incluem modelos para apoiar coordenação, percepção, serviços e modelos específicos de aplicações.

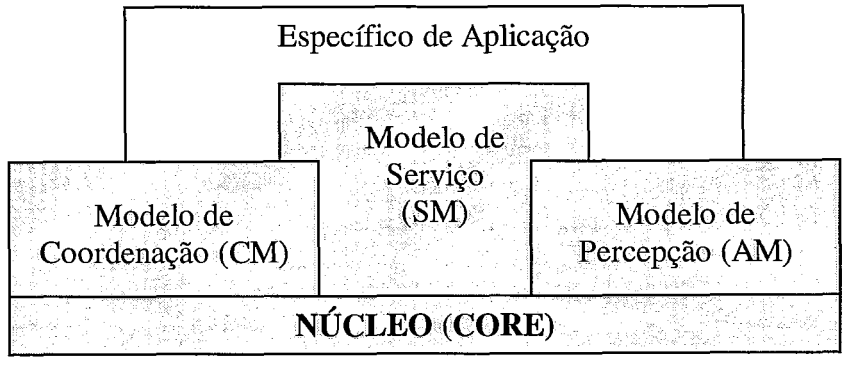

Figura 3.4 – CMM: *Core* mais Extensões

O Modelo de Coordenação (CM) fornece primitivas para coordenação de participantes e para encenação de processos. O Modelo de Serviço (SM) suporta interface

de serviço, atividades de serviços e encapsulamento de processos relacionados e contratos. O Modelo de Percepção (AM) provê o monitoramento de eventos relacionados aos processos. Sobre esses modelos podem ser definidas extensões específicas de aplicações.

➢ Arquitetura e Implementação do CMI

O sistema CMI implementa o CMM através de uma federação de sistemas comerciais e componentes próprios. A Figura 3.5 ilustra a arquitetura do CMI.

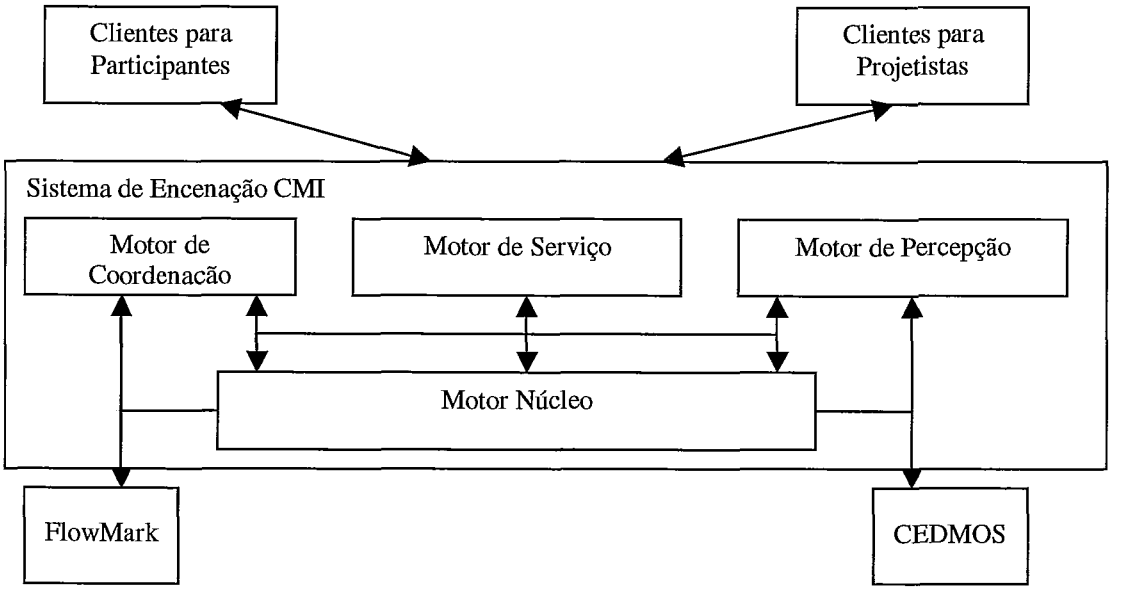

Figura 3.5 – Arquitetura do CMI

Cada um dos modelos do CMM (CORE, CM, SM e AM) é implementado por um motor na arquitetura. O CMI utiliza um WfMS comercial (IBM FlowMark) e um sistema de gerenciamento de eventos próprio (CEDMOS). Além disso, existem ferramentas clientes para que projetistas e participantes dos processos acessem o sistema.

3.3. Conclusões Preliminares

Este capítulo apresentou alguns dos problemas mais comuns no domínio de integração de processos de negócios em empresas virtuais. Foram levantados problemas referentes a aspectos tecnológicos, formas de definição de processos, autonomia, privacidade e independência de processos, problema relacionados à utilização de serviços e monitoramento, controle e sincronização de instâncias de processos.

Alguns sistemas foram propostos no meio acadêmico, com o intuito de procurar resolver ou minimizar esses problemas. Alguns deles foram apresentados nesse capítulo: o sistema WISE, o projeto CrossFlow e a infra-estrutura CMI. No entanto, nenhum deles é bem sucedido na intenção de resolver um conjunto abrangente dos problemas.

No próximo capítulo (Capítulo 4), será mostrada uma proposta de solução para os problemas apresentados nesse capítulo.

4 – Proposta de Apoio à Integração de Processos de Negócios

Nesse capítulo será mostrado um enfoque de solução para o problema da integração de processos entre empresas, visando resolver ou minimizar alguns dos problemas vistos no Capítulo 3, tais como autonomia, heterogeneidade e privacidade de processos. Serão apresentados os principais conceitos do enfoque de solução proposto, assim como requisitos para a construção de uma infra-estrutura que suporte esses conceitos.

Algumas abordagens, encontradas na literatura, propõe um sistema de *workflow* em que os processos que se deseja integrar precisam ser definidos e adaptados no novo sistema para que seja realizada essa integração. Muitas vezes, os processos internos da organização ficam dependentes dos demais processos, o que atrapalha sua autonomia.

Essa proposta, no entanto, tem como característica uma abordagem de preservação de processos internos, em uso nos sistemas de *workflow* das empresas, mantendo-os o mais intactos possível. O enfoque proposto prevê o gerenciamento do processo virtual, a abstração dos processos internos em serviços e a coordenação, sincronização, monitoramento e controle sobre esses serviços.

4.1. Empresas Virtuais

O item principal elaborado por esta dissertação é a construção de uma infra-estrutura que permita às empresas realizarem a chamada empresa virtual, isto é, permitir às empresas definirem e executarem processos virtuais. O estabelecimento de uma empresa virtual é feito por meio de um contrato, que regulamenta sua existência e é especificado de comum acordo pelas organizações envolvidas.

Processos virtuais podem ser definidos, informalmente, como processos que extrapolam os limites de uma só empresa, englobando produtores e consumidores de serviços em casos mais simples ou até mesmo intricados processos de negócios em que várias empresas realizam diversas atividades, cooperando para atingirem um objetivo comum de maior valor agregado (a entrega de um serviço ou produto a clientes).

Podem-se considerar dois modelos de empresas virtuais. No primeiro modelo, o processo é de responsabilidade comum de todas as empresas que constituem a empresa virtual, enquanto no segundo uma única empresa é dona do processo virtual, que contém atividades realizadas por outras empresas, "terceirizadas".

- Empresa virtual distribuída

Neste modelo as empresas se agrupam para oferecer um serviço com mais valor agregado do que qualquer uma delas poderia oferecer individualmente. É definido de comum acordo quais empresas participarão da chamada empresa virtual. Após essa definição, é definido o processo virtual (de posse da empresa virtual, isto é, de todas as empresas participantes), que visa estabelecer as regras de coordenação de trabalho entre as empresas e a geração de documentos e produtos intermediários. Essas empresas virtuais surgem a partir de uma necessidade ou oportunidade de mercado ou ainda de uma intenção de se antecipar ao mercado no lançamento de um serviço ou produto inovador e único. Seu tempo de duração é bastante variável, podendo ir de poucas semanas até vários anos. Após este período, a empresa é dissolvida.

- Empresa virtual centralizada

Este modelo acontece quando uma determinada empresa resolve "terceirizar" partes do seu serviço ou desenvolvimento de seu produto. Com a tendência das empresas de focarem seus negócios no seu *core business*, esse modelo é cada vez mais aplicado. Nesse modelo, o processo virtual, que contém partes realizadas internamente e partes realizadas por outras empresas, é de posse de uma só empresa, vista aqui como empresa contratante. A característica mais marcante desse processo é a possibilidade de escolha dinâmica das empresas que realizarão as partes "terceirizadas", dentro de um conjunto estabelecido. Essa escolha pode ser realizada por um executivo, que decide subjetivamente qual empresa pode melhor atender o serviço, ou automaticamente, baseado em critérios como qualidade e preço do serviço, de acordo com o contexto do momento. Essa empresa virtual, geralmente, dura o tempo em que a empresa oferecer tal serviço/produto.

4.2. Serviços como Abstrações de Processos

Segundo GREFEN et al. (2000), em empresas virtuais, um parceiro não requer detalhes operacionais de outro parceiro. Em vez disso, uma abstração bem definida de sua operação deve ser usada para se obter uma visão efetiva tanto do processo quanto dos dados. Como parceiros numa organização virtual muitas vezes possuem diferentes plataformas de TI, existe um ambiente heterogêneo (ver Capítulo 3 para uma visão mais detalhada desse problema). Essa heterogeneidade deve ser endereçada pela abstração dos detalhes técnicos dos parceiros. Para ambas as razões, as interações entre organizações devem ser definidas não em termos de seus sistemas de *workflow* (ou mesmo de processos), mas num nível de abstração acima desses sistemas.

Esse nível de abstração é o nível de serviços. Um serviço constitui uma tarefa semanticamente bem definida que é oferecida por um provedor de serviço (GEORGAKOPOULOS et al., 1999a).

A proposta parte do princípio de que cada empresa já possui seus sistemas comercias de gerenciamento de *workflow* em operação, com diversos processos de negócios definidos e sendo executados neles (ver Seção 4.3). De forma a manter a autonomia e independência de processos, a infra-estrutura tem como requisito permitir a utilização desses processos sem ser preciso alterá-los. Isso será feito desenvolvendo uma camada de gerenciamento de serviços sobre a camada de processos rodando no sistema comercial de gerência de *workflow*. A integração se dá, então, na camada de gerenciamento de serviços, o que permite deixar os processos sendo executados nos WfMSs intactos e, consequentemente, independentes e autônomos (Figura 4.1). Ao mesmo tempo, os processos internos poderão ser alterados sem que isso afete os serviços oferecidos às demais empresas, ou mesmo os que estão sendo executados no momento.

Portanto, para deixar independente o oferecimento de um serviço de sua modelagem e execução na forma de processo interno sendo encenado no WfMS, cada processo interno candidato a ser oferecido externamente terá uma representação como serviço e vice-versa, isto é, há uma equivalência entre serviço e processo. Desse modo, os processos internos ficam totalmente isolados do mundo externo e as empresas se conhecerão e se comunicarão através somente das interfaces dos serviços. O acompanhamento e controle dos serviços

serão feitos através de operações definidas na interface do serviço e verificação de estados de execução, conforme será visto na Seção 4.4.

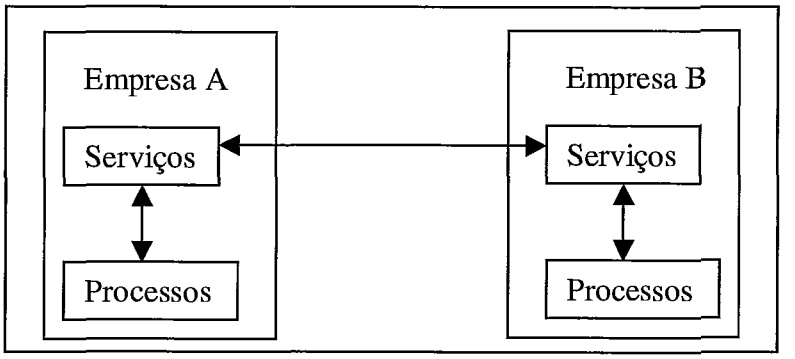

Figura 4.1 - Integração no Nível de Serviços

A camada de processos é passiva em relação à camada de serviços. Isto é, a camada de serviços "conhece" e interfere na camada de processos, mas o contrário não é válido. Isso reforça a idéia de manter os processos internos intactos e independentes. Na camada de processos, tudo acontece como se não existisse a camada de serviços nem a infra-estrutura de integração, garantindo assim a independência e autonomia desses processos. A camada de serviços está constantemente monitorando a camada de processos e verificando mudanças no fluxo de trabalho entre atividades. A determinadas mudanças (ou conjunto de mudanças) ocorridas nas atividades correspondem mudanças nos estados do serviço.

Um aspecto muito importante das relações de negócios entre empresas refere-se ao registro de dados das instâncias de serviços contratados, porque permite realizar auditorias, com o intuito de verificar se o serviço foi executado de acordo com o contrato legal. Esse registro pode servir como prova no caso de disputas legais. Logo, é importante ser minucioso no registro de dados referentes à execução de serviços, armazenando todos os eventos de destaque que aconteçam. Outra vantagem ainda desse registro é permitir a melhoria contínua desses processos, através da reengenharia dos mesmos.

4.3. Utilização de WfMSs Comerciais

A infra-estrutura utilizará como base para definição e encenação de processos sistemas comerciais de gerenciamento de *workflow* (WfMSs). Isso se deve a três motivos principais.

Primeiro, porque esses sistemas atendem satisfatoriamente às necessidades da infra-estrutura para definição e encenação de processos. Quanto à encenação de processos, os

processos que serão integrados no processo virtual já estarão em uso dentro da empresa, portanto, não será necessário fazer qualquer tipo de alteração neles e, consequentemente, deduz-se que esses sistemas são satisfatórios para esse fim. A gerência dos processos virtuais (e a coordenação dos processos que o compõe) será feita numa camada acima, a camada de serviços.

Portanto, seria desnecessário "reinventar a roda", isto é, implementar toda a funcionalidade de definição de processos (definição de fluxos de trabalho, criação de estruturas de dados, criação de papéis e pessoas, associação de papéis ou pessoas a atividades, etc.) e tarefas intrínsecas à encenação de processos (coordenação de fluxo de atividades, gerência de recursos, gerência de listas de trabalho, etc.) uma vez que podemos contar com esses recursos em ferramentas disponíveis no mercado.

O segundo motivo é baratear o custo da implantação da infra-estrutura. Segundo LAZCANO e ALONSO (2001), a maior dificuldade para a criação e execução de um processo interorganizacional não tem origem nas grandes empresas, mas sim nas pequenas e médias empresas e sua falta de recursos computacionais. Ao utilizar sistemas comerciais, já em uso nas empresas, retira-se um custo extra do orçamento desta infra-estrutura, referente à instalação, implantação e configuração da mesma. São eliminados também os custos de aprendizado de uma nova ferramenta e de sua administração e todo o trabalho de (re)definição de processos em um novo sistema.

A infra-estrutura deverá conter uma camada de gerenciamento abstrato de *workflow*, que corresponde a um WfMS virtual, semelhante ao conceito de "máquina virtual" encontrado em linguagens de programação. Todas as chamadas ao sistema de *workflow* são feitas sobre essa camada, que isola o sistema comercial propriamente dito. Assim, qualquer WfMS comercial poderá ser usado, desde que exista uma implementação da "máquina virtual" para esse sistema, o que evita a obrigatoriedade de se adotar (quer dizer, comprar) um determinado WfMS comercial. O uso do WfMS abstrato e sua comunicação com o restante da arquitetura da infra-estrutura será mais bem detalhado nos próximos capítulos.

O terceiro motivo é afetar o menos possível o modo como as empresas trabalham. Caso contrário, seria necessário redefinir os processos usados pela empresa em seus WfMSs na nova infra-estrutura. Isso causaria também mudanças na maneira habitual das pessoas trabalharem, causando assim perda de produtividade.

O esquema de dados (e respectivos dados) utilizado pela infra-estrutura será armazenado num sistema gerenciador de banco de dados. Todas as operações sobre o banco de dados serão feitas através de ODBC (ODBC, 1999), portanto qualquer DBMS comercial poderá ser utilizado.

4.4. Monitoramento e Controle de Serviços

Uma vez contratado o serviço, seguindo o ciclo de vida de um comércio eletrônico entre empresas (*Business-to-Business*) (SCHMID, 1997), começa-se a fase de execução e monitoramento do serviço, que dura até a sua conclusão.

O acompanhamento e controle de integração entre empresas se dão através de serviços, ou seja, uma empresa não conhece os processos internos da outra empresa, que é acessível somente através do serviço contratado.

O acompanhamento, ao invés de ser feito através do fluxo de atividades do processo, será feito através da noção de estados do serviço, baseando-se no trabalho de (GEORGAKOPOULOS, 1999a). Os modelos e sistemas tradicionais de *workflow* (WfMC-TC-1011, 1999, OMG-WMF, 2000, IBM-MQWF, 2001a) contemplam somente estados genéricos pelos quais uma instância de processo pode passar, baseados na definição genérica criada pela WfMC (WfMC-TC-1011, 1999). Os estados definidos pela WfMC são: "pronto", "executando", "suspenso", "completo" (finalizado com sucesso) e "terminado" (finalizado por alguma falha ou por cancelamento explícito do usuário). Porém, esses estados são muito vagos e insuficientes para se acompanhar o andamento de um serviço.

A infra-estrutura oferece esses estados definidos pela WfMC como estados básicos, mas propicia o conceito de estados específicos de aplicação, que são estados que refletem de forma mais clara o andamento do serviço, relacionados com características inerentes ao serviço. Por exemplo, o estado genérico "executando" pode ser refinado em dois novos estados: "provendo serviço" e "registrando cobrança" (Figura 4.2). Quando um estado é substituído, todas as suas transições devem ser substituídas por transições envolvendo os subestados. Assim, mantém-se coerência entre as transições de estados que já existiam. Na Figura 4.2, a transição entre "pronto" e "executando" foi substituída por outra entre "pronto" e "registrando cobrança". A transição entre "executando" e "terminado" foi

substituída pela transição entre "registrando cobrança" e "terminado". Por fim, a transição entre "executando" e "completo" foi substituída pela transição entre "registrando cobrança" e "completo".

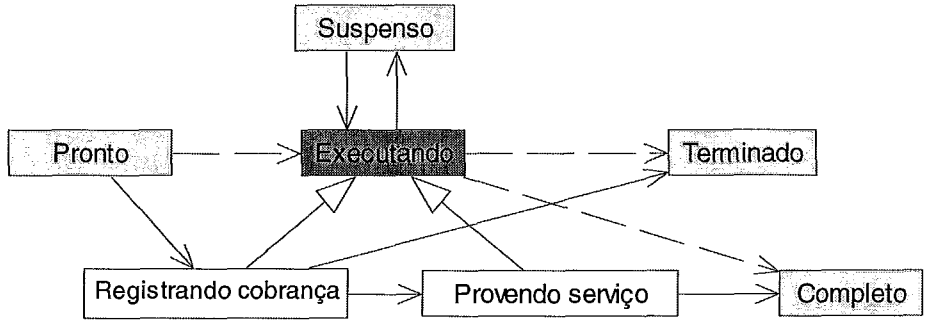

Figura 4.2 – Criando Novos Estados

Cada serviço está associado a um esquema de estados, onde são definidos os estados pela qual o serviço pode passar e suas respectivas transições, além das operações, responsáveis por realizar as transições entre estados e que podem ser invocadas pelo contratante do serviço. De forma a ser consistente com o modelo de estados do WfMC, todo esquema de estados especializa as características (estados e transições) do modelo do WfMC. De forma similar, um esquema de estados específico de aplicação pode especializar outro esquema de estados, definindo assim uma hierarquia de esquemas de estados.

A forma primordial de acompanhamento do serviço é a consulta ao estado em que se encontra o serviço. Outra forma de acompanhamento do andamento do serviço, passiva, é através de notificações. Ao estabelecer o contrato, a empresa contratante pode definir em que tipos de eventos está interessada em tomar conhecimento. Esses eventos estão relacionados à entrada (ou saída) de estados. Sempre que um evento acontecer, uma notificação é enviada automaticamente à empresa interessada no evento.

O controle é feito invocando operações sobre o serviço. Cada serviço possui um conjunto de operações, que permitem criar, iniciar, suspender, retomar, completar e terminar atividades e podem ser invocadas pelas organizações clientes dos serviços. Nos modelos tradicionais de *workflow*, as operações são implícitas. Isto é, oferece-se somente a possibilidade de executar ações pré-determinadas sobre instâncias de processos, de maneira restrita. O modelo proposto nesse trabalho utiliza o conceito explícito de operação.

Todas as operações causam transições entre estados do serviço. Assim, ao serem criados novos estados para refletir o andamento de aplicações, podem-se criar operações

para causar transições envolvendo esses novos estados. Como os serviços são mapeados para processos internos, uma mudança de estado na camada de serviços implica uma mudança no fluxo de atividades do processo interno.

4.5. Mapeamento entre Serviços e Processos Internos

Um serviço possui duas "faces" distintas: uma interface e uma implementação. A interface é uma especificação puramente declarativa, semelhante ao conceito equivalente encontrado na orientação a objetos, como por exemplo, o conceito de declaração de objetos através da linguagem IDL, definido pela OMG (OMG, 1998). Em outras palavras, uma interface é uma declaração de operações abstratas que podem ser invocadas pelo contratante do serviço. Além das operações, a definição de interface apresentada aqui também engloba um esquema de estados de serviços e transições entre eles, a fim de permitir o monitoramento do serviço e seu controle.

Os estados de serviços, encontrados na interface de um serviço, são determinados pelos estados das atividades do processo interno mapeado por ele. Foi visto que a camada de processos é passiva em relação à camada de serviços. A fim de fornecer informações corretas sobre o estado de um serviço, o sistema deverá monitorar constantemente a instância de processo correspondente. Ao ocorrerem mudanças na instância de processo, pode ocorrer mudança no estado do serviço. A forma como as mudanças na instância de processo afetam o estado do serviço será definida numa linguagem lógica. Por exemplo, poder-se-ia dizer que um serviço entra no estado "executando serviço técnico" quando a atividade "registrando solicitação" estiver completa E a atividade "cobrando serviço" ainda não tiver sido executada.

A linguagem que será adotada para especificação das interfaces de serviços baseia-se nas especificações dos padrões para descrição de *Web Services* WSDL (W3C-WSDL, 2001) e no padrão de comércio eletrônico ebXML (EBXML-CPP, 2001), que por sua vez utilizam a linguagem XML (W3C-XML, 2000). Portanto, as interfaces de serviços também serão declaradas em XML, beneficiando-se das características dessa linguagem, como flexibilidade, extensibilidade e portabilidade.

A implementação do serviço visa a real execução do mesmo, de acordo com a interface definida. Enquanto a interface é uma especificação meramente declarativa, a

implementação de um serviço precisa ser codificada numa linguagem de programação, de forma a fornecer o serviço em termos do processo interno que roda no WfMS da empresa. Ou seja, a cada operação executada sobre um serviço corresponderá uma série de ações a serem executadas sobre as atividades do processo interno. Uma interface de serviço pode possuir diversas implementações, para que se possam definir diversas maneiras de se realizar um serviço (por exemplo, com parâmetros de qualidade de serviço distintos).

A implementação do serviço é feita pelo provedor do serviço. Os clientes do serviço somente "enxergam" sua interface, sem conhecer sua implementação. Como a comunicação entre serviços será feita utilizando-se protocolos neutros de comunicação, segundo o conceito de *web services*, a implementação do serviço não precisa se ater a detalhes de comunicação.

Já que as interfaces de serviços contêm uma definição de esquema de estados, as interfaces também podem especializar outras interfaces, o que representa um mecanismo de refinamento progressivo de serviços.

4.6. Definição e Encenação de Processos de Negócios Virtuais

Segundo JACOBSON (1995), "um processo de negócios é o conjunto de atividades internas desempenhadas para servir um cliente". Essa definição simples vale para processos de negócios modelados e desempenhados dentro de uma mesma organização.

Inspirado nessa definição, pode-se definir um processo de negócios virtual como um conjunto de atividades virtuais desempenhados para servir um cliente. "Processo de negócios virtual" e "processo virtual" são utilizados implicitamente como equivalentes. Uma atividade virtual, por sua vez, pode ser tanto uma atividade interna (local) quanto um serviço desempenhado por outro departamento ou organização (remotamente).

Do ponto de vista da modelagem e execução de processos em WfMSs, temos a definição de processo da WfMC: "a representação de um processo de negócios numa forma que suporta manipulação automática, como modelagem, ou execução por um sistema de gerenciamento de *workflow*. A definição do processo consiste de uma rede de atividades e seus relacionamentos, critérios para indicar o início e fim de um processo e informação sobre as atividades individuais, como participantes, aplicações de TI e dados associados, etc." (WfMC-TC-1011, 1999).

Vendo essa mesma definição sob a ótica de processos virtuais, devemos considerar que a definição do processo consiste de uma rede de atividades virtuais e seus relacionamentos. A definição de atividade virtual referente a WfMSs é a de uma atividade a ser executada em algum sistema de gerenciamento de *workflow* dentro da empresa ou um serviço sendo executado remotamente em outro departamento, ou mesmo em outra organização.

Com essas definições, já podemos ter uma idéia razoavelmente clara de como compor e coordenar as diversas atividades e serviços que constituem um processo que engloba mais de uma empresa (processo virtual). É importante ressaltar que cada atividade no processo virtual corresponde, na verdade, a um processo rodando em algum WfMS local ou remoto.

Ao inserir uma atividade no processo virtual, declara-se a que tipo de esquema de estados a atividade está associada, da mesma forma como se associa uma variável a um tipo, em linguagens de programação, quando a variável é declarada. Essas são atividades virtuais. Desse modo, na hora da modelagem não é preciso definir ainda como a atividade será executada, apenas que tipo de atividade deve ser realizada. Somente na hora da execução do processo será preciso determinar o serviço que implementará aquela atividade virtual. Essa definição pode ser feita de forma automática, baseando-se em heurísticas para escolha da atividade mais adequada naquele contexto (decidida a partir de critérios como custo da atividade, tempo de execução, etc.) ou manualmente por um executivo, responsável pelo processo. É importante salientar que o serviço deve ser do mesmo tipo de esquema de estados ou de um tipo filho (segundo o conceito de herança na orientação a objetos) que a atividade virtual declarada. Dessa forma, é endereçado o problema de escolha dinâmica de serviços.

No entanto, é oferecida a possibilidade de se definir *a priori* o serviço que deverá ser executado naquela atividade virtual, por *default*, facilitando assim a escolha em tempo de execução.

Distinguem-se como serviços dois tipos básicos: serviços locais, que podem estar definidos em qualquer WfMS comercial em uso na empresa e serviços remotos. Para o primeiro tipo, o projetista do processo deve indicar a que processo corresponde a atividade

e em que WfMS ele será executado. No segundo tipo, é assinalado o serviço remoto a ser executado e a que organização ele pertence.

As transições entre as atividades são determinadas de maneira semelhante à modelagem de processos tradicionais, com algumas diferenças: no modelo tradicional de *workflow*, quando há uma transição entre duas atividades, assume-se implicitamente como critério para o início da segunda atividade o fim da primeira. Ou seja, a primeira atividade deve entrar no estado "terminado" ou "finalizado", segundo a máquina de estados definida pela WfMC, para que a segunda possa ser iniciada. No modelo apresentado aqui, no entanto, pode-se definir explicitamente como critério para início de uma atividade um estado qualquer, que pode ser um dos estados genéricos baseados na WfMC ou um estado específico de aplicação. Caso não seja definido explicitamente nenhum critério de início, utiliza-se o mesmo critério dos modelos tradicionais.

Nos modelos tradicionais, ao ocorrer uma transição entre suas atividades, a atividade destino sempre é posta no estado "executando". No entanto, na transição para uma atividade virtual definida nesse modelo, especifica-se também o estado em que deve entrar a atividade destino da transição. Assim, podemos definir formalmente uma transição entre atividades virtuais da seguinte forma: $(ativ_{orig}, estado_{orig}) \rightarrow (ativ_{dest}, estado_{dest})$.

Esse novo conceito de transição entre atividades virtuais permite a realização da coordenação e sincronização entre processos e serviços, ou mesmo entre serviços, no nível do processo virtual. Uma atividade só pode ser iniciada a partir de um comando explícito do processo virtual, o que permite a este coordenar e sincronizar atividades de maneira simples e eficiente. Todas as formas de sincronização previstas na interface 4 da WfMC (WfMC-TC-1012, 1999) podem ser realizadas com a infra-estrutura proposta nesse trabalho.

4.7. Conclusões Preliminares

Esse capítulo apresentou uma proposta de solução para o problema de integração de processos de negócios entre empresas. Foram discutidos os principais aspectos da solução, visando amenizar ou solucionar os problemas discutidos no Capítulo 3.

Os conceitos mais importantes tratados pela proposta foram: empresas virtuais, serviços; contratos; uso de WfMSs comerciais; tipos de esquema de estados; estados

específicos de aplicação; herança de esquemas de estados; processos virtuais; declaração de atividades virtuais; escolha dinâmica de serviços; coordenação e sincronização de atividades.

Esses conceitos servirão de guia para a definição da arquitetura e requisitos da infra-estrutura. Nos próximos capítulos serão apresentados a arquitetura da solução e seu detalhamento, de forma a mostrar como os conceitos aqui apresentados foram implementados no protótipo da infra-estrutura.

Entre as características de destaque dessa proposta, em relação a outros sistemas propostos, estão:

- implementação do serviço está no provedor e não no cliente. O cliente pode utilizar o serviço sem se preocupar em antes adaptá-lo.
- há possibilidade de se utilizar diversos WfMSs
- não exige que os processos definidos nos WfMSs sejam remodelados para serem integrados, evitando um custo extra e perda de produtividade
- processos virtuais englobam serviços, sejam externos ou internos
- utiliza padrões neutros de *web* (*web services*): abstrai protocolos de comunicação
- para os participantes dos processos, é como se não houvesse integração. Seu modo de trabalho continua como era antes, inclusive em relação a ferramentas utilizadas
- processos são independentes: a coordenação de serviços é feita no nível de serviço e não nos processos internos
- não mistura atividades comuns com serviços, o que complica o projeto, implementação e desempenho do sistema e compromete a consistência dos processos internos
- permite mapeamento mais rico de serviços a processos internos, através de condições complexas expressas na linguagem
- serviços recebem notificações de mudanças, em vez de precisarem ficar constantemente chamando operações em outros serviços, a fim de monitorá-los
- ao permitir a definição de serviços internos, também facilita a integração de aplicações dentro da empresa (*Enterprise Application Integration – EAI*)

5 - Modelo de Dados e Arquitetura da Infra-Estrutura

Esse capítulo tem como objetivo apresentar o modelo de dados que servirá como base para armazenamento e manipulação das informações utilizadas pela infra-estrutura. Esse modelo pode ser entendido como um meta-modelo que oferece as primitivas para definição de aplicações em empresas virtuais. Também será apresentada a arquitetura da infra-estrutura, com seus componentes principais. Tanto o modelo de dados quanto a arquitetura procuram espelhar e viabilizar os conceitos discutidos na proposta de solução apresentada no Capítulo 4. O modelo de dados é apresentado na seção 5.1 e a arquitetura na seção 5.2.

5.1. Modelo de Dados

O modelo de dados da infra-estrutura representa conceitualmente as informações pertinentes ao domínio do problema abordado e visa possibilitar o armazenamento dessas informações. Portanto, pode ser considerado como um meta-modelo para definição de aplicações. A Figura 5.1 apresenta as cinco categorias em que as classes do modelo de dados foram divididas. Os diagramas apresentados nessa seção estão representados na linguagem UML (FURLAN, 1998).

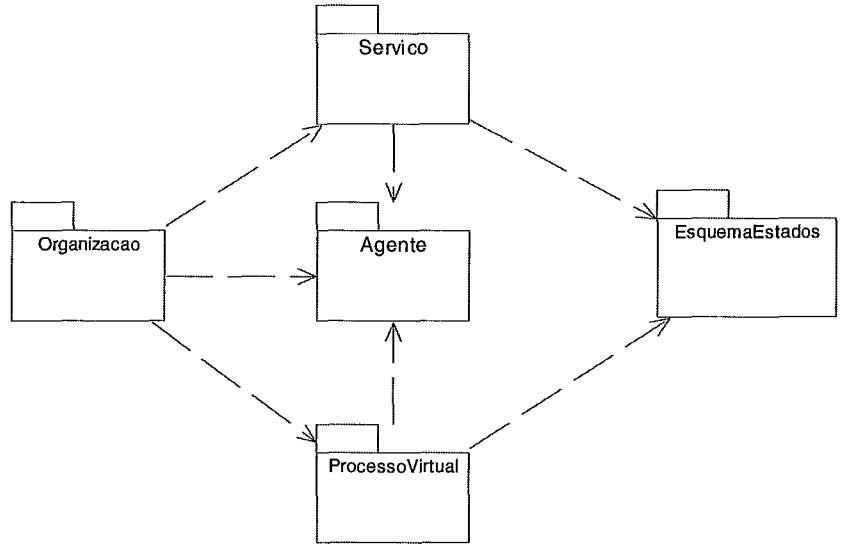

Figura 5.1 – Modelo de Dados da Infra-Estrutura – Diagrama de Categorias

As categorias desse modelo de dados serão descritas nas próximas seções segundo o padrão encontrado no trabalho de DIAS (1998). Esse padrão contém os seguintes itens:

- **Objetivo**: apresenta o objetivo da categoria, fornecendo uma idéia geral do tipo de serviço relacionado com as classes da categoria;
- **Categorias Relacionadas**: apresenta as categorias das quais a categoria descrita depende;
- **Diagrama**: apresenta o diagrama de classes dessa categoria. As classes que pertencem à categoria estão representadas com fundo branco, enquanto as classes de outras categorias com as quais as classes dessa categoria se relacionam estão representadas com fundo cinza, para diferenciá-las.
- **Descrição sumária das classes**: apresenta uma breve descrição de todas as classes da categoria, incluindo o objetivo de cada uma dentro desse contexto. Pode conter também outros diagramas a fim de ilustrar aspectos referentes à descrição das classes.

5.1.1. Categoria Agente

Objetivo: Permitir a representação e armazenamento de informações relativas a agentes básicos que podem executar implementações de serviços.

Categorias Relacionadas: Nenhuma

Diagrama:

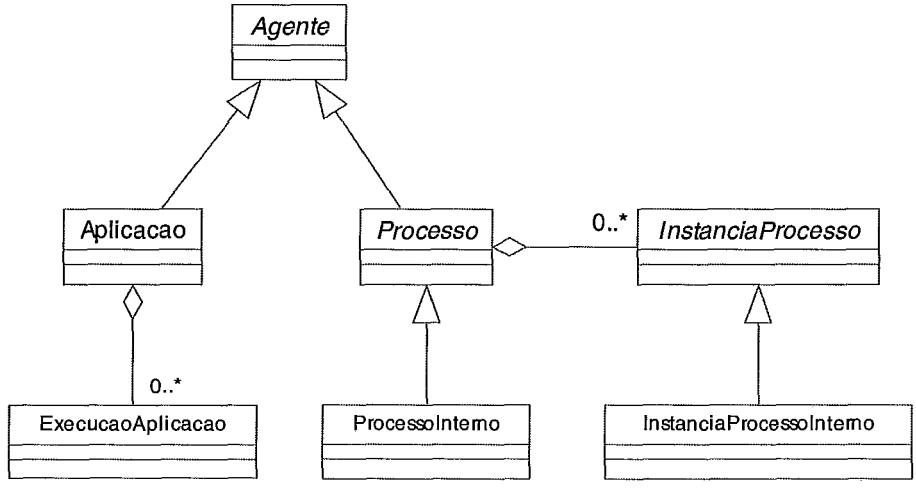

Figura 5.2 – Categoria Agente

Descrição sumária das classes:

A classe Agente representa uma entidade que pode vir a executar um serviço. Distinguem-se dois tipos de agente: Aplicação (classe Aplicacao), que armazena informações relativas a programas que podem ser executados para desempenhar uma tarefa e Processo, que são definições de fluxo de trabalho.

A classe Aplicacao representa um sistema de informação que pode ser utilizado, através da abstração de serviço (ver seção 5.1.3), para a realização de tarefas. Assim, podem-se aproveitar sistemas em uso nas corporações (sistemas legados), pois geralmente representam um grande investimento feito em tecnologia da informação por essas empresas, os quais muitas vezes elas não querem descartar ou substituir. Entre as informações armazenadas na classe Aplicacao estão um identificador, nome, descrição, endereço físico (*path*), parâmetros de invocação e, opcionalmente, *login* e senha de acesso para as aplicações que requerem validação e autorização de usuário.

As aplicações podem estar encapsuladas em algum modelo de objetos distribuídos, como CORBA, Enterprise Java Beans (EJB) ou Microsoft COM/DCOM, o que facilita o seu uso por sistemas orientados a objetos baseados em serviços, como o proposto neste trabalho.

Muitas vezes, as aplicações são *stand-alone* e, nesse caso, são invocadas de forma pontual, isto é, sem integração explícita com outras ferramentas. Contudo, as aplicações utilizadas na infra-estrutura podem também estar participando de um fluxo de trabalho definido pela organização, interagindo com outras aplicações através de parâmetros de processo ou compartilhamento de dados. Um método de coordenação de aplicações através de fluxos de trabalho, gerenciados por sistemas de *workflow*, pode ser visto em outra dissertação (VINCENT, 2002).

A classe Processo é uma classe abstrata que representa uma definição de processo, ou fluxo de trabalho. Um objeto Processo registra todas as instâncias de processo (classe InstanciaProcesso) criadas a partir de sua definição, em tempo de execução. Nessa categoria foram definidas ainda duas especializações de Processo e InstanciaProcesso: ProcessoInterno e InstanciaProcessoInterno, respectivamente. A classe ProcessoInterno tem por objetivo registrar processos que são definidos exclusivamente dentro do contexto de uma só organização e têm uma equivalência direta com processos definidos em sistemas de gerenciamento de *workflow*. Para tanto, ela guarda basicamente um "ponteiro" para a

localização dessa definição no WfMS, pois seria redundante armazenar toda a definição novamente na infra-estrutura. A classe InstanciaProcesso representa as instâncias de processo que são executadas nesses sistemas, a partir das definições encontradas em Processo.

5.1.2. Categoria Esquema de Estados

Objetivo: Permitir a definição de um tipo de esquema de estados, isto é, uma definição de máquina de estados, que contém estados, as transições existentes entre eles e as operações que causam essas transições. As informações definidas nessa categoria servirão para indicar tipos de serviços e atividades em outras categorias.

Categorias Relacionadas: Nenhuma

Diagrama:

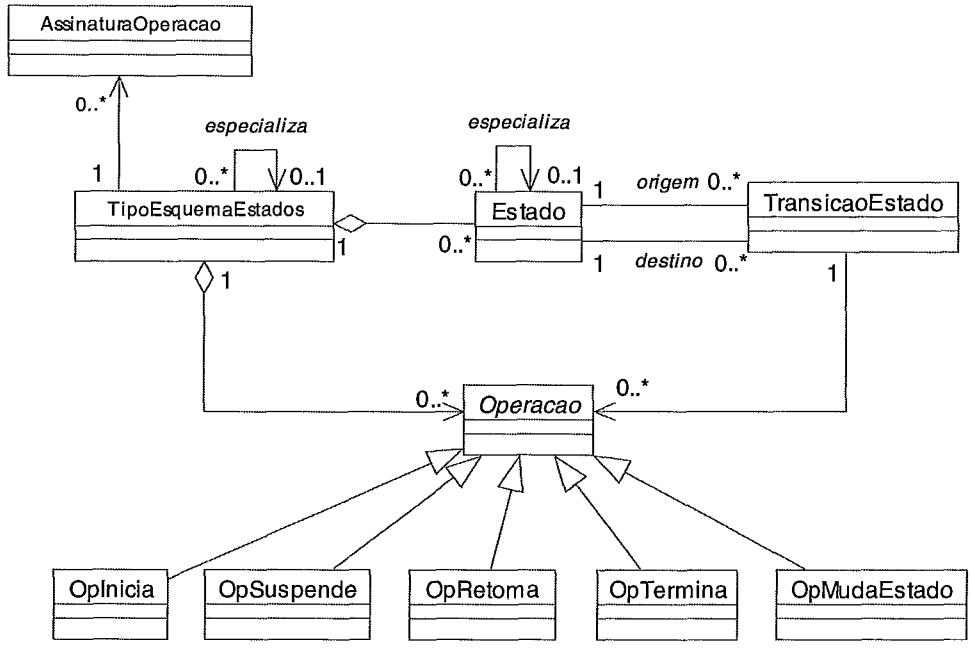

Figura 5.3 – Categoria Esquema de Estados

Descrição sumária das classes:

A classe TipoEsquemaEstados é a principal classe dessa categoria e representa um meta-modelo para definição de uma máquina de estados. Um tipo de esquema de estados (TipoEsquemaEstados) contém estados e operações, que podem ser invocadas sobre esses estados.

Cada novo serviço criado na infra-estrutura poderá utilizar um tipo já existente ou definir o seu próprio tipo. Como na orientação a objetos, onde um tipo (classe) pode especializar outro, herdando suas características, no meta-modelo definido nesse trabalho também um TipoEsquemaEstados pode especializar outro. Através desse mecanismo podem-se criar serviços que refinam outro serviço.

Um TipoEsquemaEstados contém uma lista de assinaturas de operações. Uma assinatura de operação (classe AssinaturaOperacao) contém informações como o nome da operação, parâmetros para chamada da operação, transição a que se refere e o caminho (*path*) da classe que a implementará. As operações definidas em um tipo poderão ser invocadas pelos clientes dos serviços que implementem esse tipo, definindo dessa forma a interface do serviço, como será descrito em mais detalhes na seção 5.1.3.

A classe Operacao permite a implementação efetiva das operações definidas para um tipo de esquema de estados (representadas em AssinaturaOperacao). A classe operação possui algumas especializações, correspondentes às operações que podem ser executadas sobre processos encontradas no *Workflow Management Facility* da OMG (OMG-WMF, 1998). As especializações são OpInicia, OpSuspende, OpRetoma, OpTermina e OpMudaEstado. A fim de se definirem operações específicas para um determinado serviço, deverão ser criadas novas classes de operações, especializando uma das classes básicas de operação definidas na infra-estrutura.

A classe Estado representa um estado em que pode se encontrar um objeto que instancia essa máquina de estados. A fim de manter coerência entre os tipos de esquemas de estados, quando há especialização entre eles, um estado pode especializar um outro estado definido no TipoEsquemaEstados pai do tipo que o contém.

A classe TransicaoEstados permite definir uma transição entre dois estados. Um objeto dessa classe registra um estado de origem e um estado de destino, sendo que um estado pode participar de diversas transições como origem ou destino de outros estados. É armazenada nessa classe também uma expressão lógica que indica a condição (composta) necessária para que se realize a transição entre estados. Por exemplo, poder-se-ia definir a seguinte expressão: "Atividade 1 está completa e Atividade 2 está suspensa ou Atividade 3 está abortada".

A implementação efetiva da expressão é feita através da operação a que o objeto TransicaoEstados está relacionado. No caso de processos, uma transição de estados invoca uma operação que pode ocasionar uma ou mais transições entre atividades do fluxo de trabalho. No caso de aplicações, a operação tomará as medidas necessárias para que a aplicação permaneça num estado coerente com o estado definido na máquina de estados.

Um tipo básico de esquema de estados é definido por *default*, que corresponde à máquina de estados para processos sugerida pela WfMC. A Figura 5.4 ilustra, por meio de um diagrama de objetos, a definição desse tipo através do objeto tipoWfMC (instância da classe TipoEsquemaEstados) e os demais objetos relacionados com ele. Todo tipo de esquema de estados criado na infra-estrutura possui como base na sua hierarquia de tipos o objeto tipoWfMC.

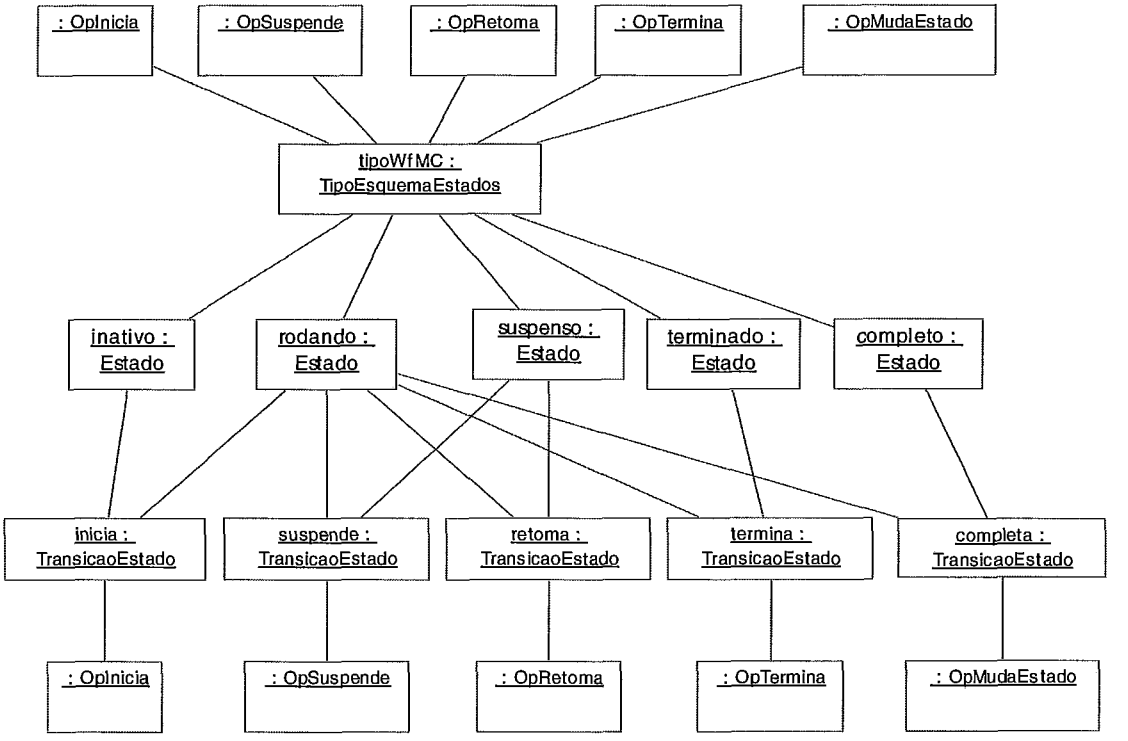

Figura 5.4 – Diagrama de Objetos para o tipo de Esquema de Estados WfMC

5.1.3. Categoria Serviço

Objetivo: Permitir a definição de serviços e relacioná-los com uma implementação, além de definir como os eventos relacionados a serviços serão registrados.

Categorias Relacionadas: Agente e EsquemaEstados

Diagrama:

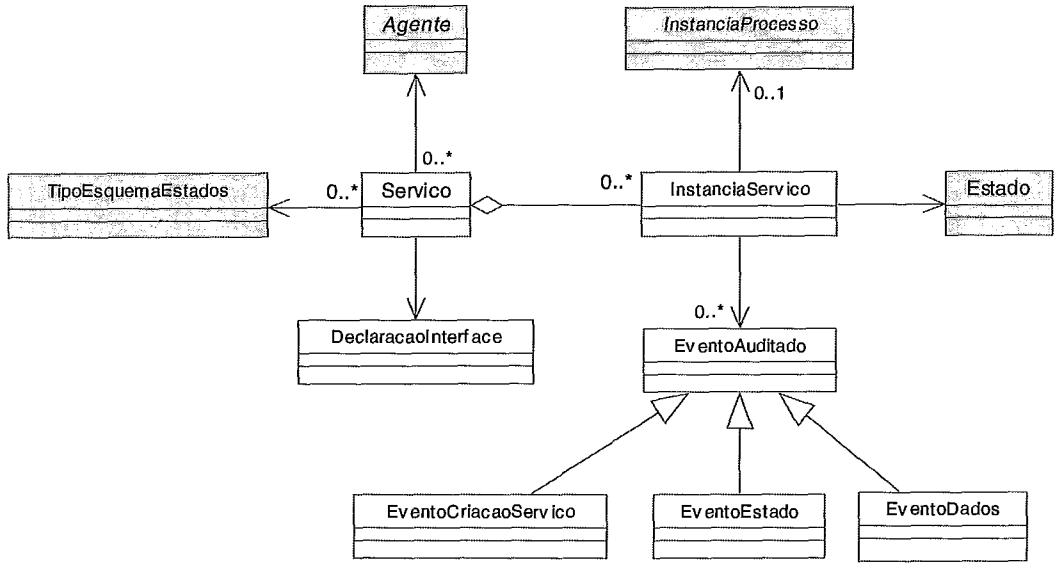

Figura 5.5 – Categoria Serviço

Descrição sumária das classes:

Um serviço é o elemento básico para a integração de empresas. A classe Servico oferece um meta-elemento para criação de serviços. Essa classe armazena informações pertinentes ao uso do serviço, como nome (como ele será invocado quando for utilizado), parâmetros de entrada e saída, resultado esperado, uma breve descrição de condições de uso e alguns critérios de qualidade de serviço (QoS). Um serviço implementa um determinado tipo de esquema de estados, representado no modelo através do relacionamento com a classe TipoEsquemaEstados da categoria EsquemaEstados, que define como o serviço poderá ser monitorado.

Um serviço possui uma declaração de interface (classe DeclaracaoInterface) onde são definidas as operações que poderão ser invocadas para monitorar e controlar o serviço. Essa declaração de interface é representada na linguagem WSDL (W3C-WSDL, 2001).

A implementação do serviço é feita através de um agente (relacionamento com a classe Agente da categoria homônima), isto é, o serviço pode encapsular uma aplicação ou um processo responsáveis por realizar a tarefa a que o serviço se propõe a oferecer.

Uma instância de serviço (classe InstanciaServico) é criada cada vez que um serviço é executado e está relacionada com a sua definição de serviço. Diversas instâncias de serviço podem ser criadas para um serviço e cada uma delas poderá ser monitorada e

controlada independentemente. Caso o agente do serviço seja um processo, a instância de serviço possuirá uma ligação com a instância de processo que será executada como sua implementação.

A fim de permitir o registro de eventos importantes gerados durante o ciclo de vida de uma instância de serviço, foram criadas a classe EventoAuditado e suas especializações: EventoCriacaoServico, EventoEstado e EventoDados, baseadas no modelo da *Workflow Management Facility*, da OMG. A classe EventoCriacaoServico armazena, entre outras informações, a data de criação da instância de serviço e parâmetros de entrada utilizados na instanciação. A classe EventoEstado armazena informações relativas a transições de estados ocorridas na execução do serviço. Por último, a classe EventosDados permite registrar eventos de mudanças de dados no âmbito da execução de uma instância de serviço.

5.1.4. Categoria Processo Virtual

Objetivo: Oferecer os elementos básicos para modelagem de processos virtuais e armazenar as informações relativas à sua definição.

Categorias Relacionadas: Agente e EsquemaEstados

Diagrama:

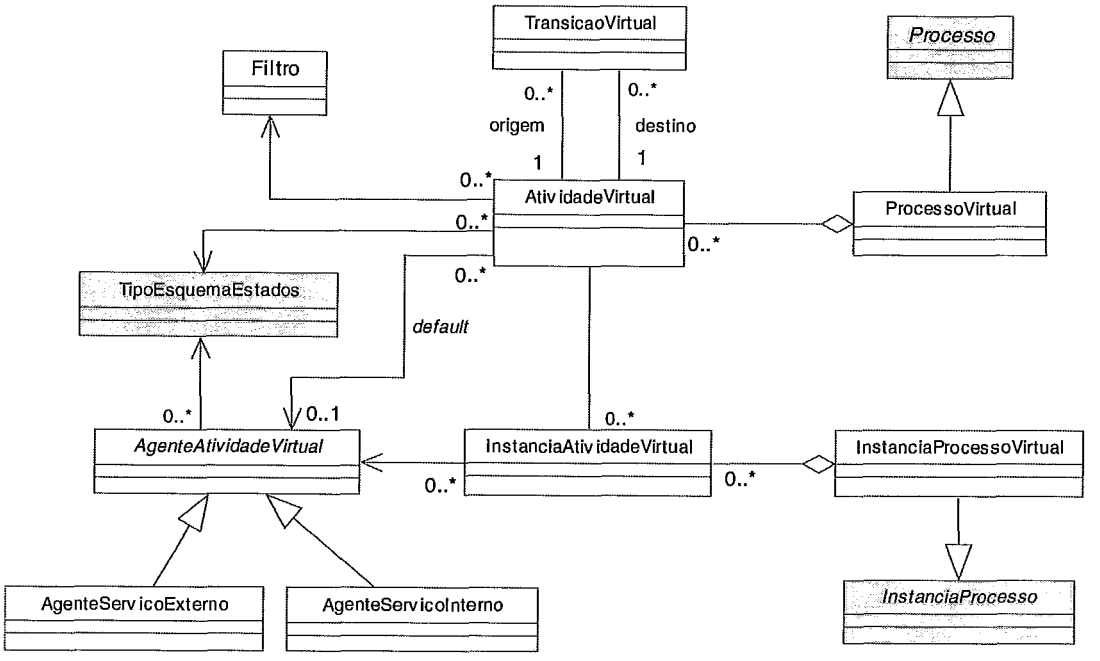

Figura 5.6 – Categoria Processo Virtual

Descrição sumária das classes:

Um Processo Virtual representa um processo definido e encenado através da composição de serviços de diversas empresas. A classe ProcessoVirtual representa essa idéia, sendo uma especialização da classe genérica Processo, da categoria homônima.

Um ProcessoVirtual é composto de unidades de trabalho, chamadas de atividades virtuais (classe AtividadeVirtual). Em outras palavras, um processo virtual é definido através da coordenação de diversas atividades virtuais. A transição entre duas atividades virtuais é definida na classe TransicaoVirtual.

Na modelagem do processo virtual, ao declarar-se uma atividade virtual não é necessário indicar *a priori* como essa atividade será realizada. Basta declarar a que tipo pertence aquela atividade, permitindo assim a escolha dinâmica de serviços em tempo de execução. Isso é expresso na ligação que relaciona AtividadeVirtual a TipoEsquemaEstados, da categoria EsquemaEstados. Um Filtro pode ser utilizado para restringir o espaço de possibilidades de tipos que poderão ser utilizados. Essa classe permite especificar também os parâmetros de qualidade de serviço (QoS) desejados para a atividade.

Há duas possibilidades de escolha de agentes para a realização de uma atividade virtual, especializações da classe Agente: agentes externos, que encapsulam serviços externos, isto é, serviços contratados de outras organizações (classe AgenteServicoExterno) e agentes internos, que encapsulam serviços que abstraem processos internos à organização (classe AgenteServicoInterno). Isso dá liberdade operacional à empresa para escolher entre realizar uma determinada atividade definida no processo virtual internamente ou "terceirizar" sua realização.

No entanto, caso o agente que realizará a atividade já esteja definido em tempo de construção, este poderá ser assinalado à atividade. É isso que expressa o relacionamento default entre as classes AtividadeVirtual e AgenteAtividadeVirtual. O agente default deve ser do mesmo tipo que a AtividadeVirtual.

Em tempo de execução, cada vez que um processo virtual for criado, a ele será associada uma nova instância de processo virtual (classe InstanciaProcessoVirtual). Essa classe é uma especialização da classe abstrata InstanciaProcesso da categoria Processo. Uma InstanciaVirtual é composta de instâncias de atividades virtuais (classe

InstanciaAtividadeVirtual), uma para cada atividade virtual definida no processo. É nesse momento que deverão ser escolhidos os agentes realizadores das atividades declaradas que não possuem um agente default. Haverá uma ligação entre elas e os agentes escolhidos.

Poderão ser escolhidos dois tipos de agentes, especializações da classe Agente: agentes externos, que encapsulam serviços externos, isto é, serviços contratados de outras organizações (classe AgenteServicoExterno) e agentes internos, que encapsulam serviços que abstraem processos internos à organização (classe AgenteServicoInterno). Isso dá liberdade operacional à empresa virtual para escolher entre realizar uma determinada atividade definida no processo virtual internamente ou "terceirizar" sua realização.

5.1.5. Categoria Organização

Objetivo: Permitir a modelagem de organizações físicas e virtuais e seu armazenamento, além de capturar a interação entre elas através do estabelecimento de contratos.

Categorias Relacionadas: Serviço, Agente e ProcessoVirtual.

Diagrama:

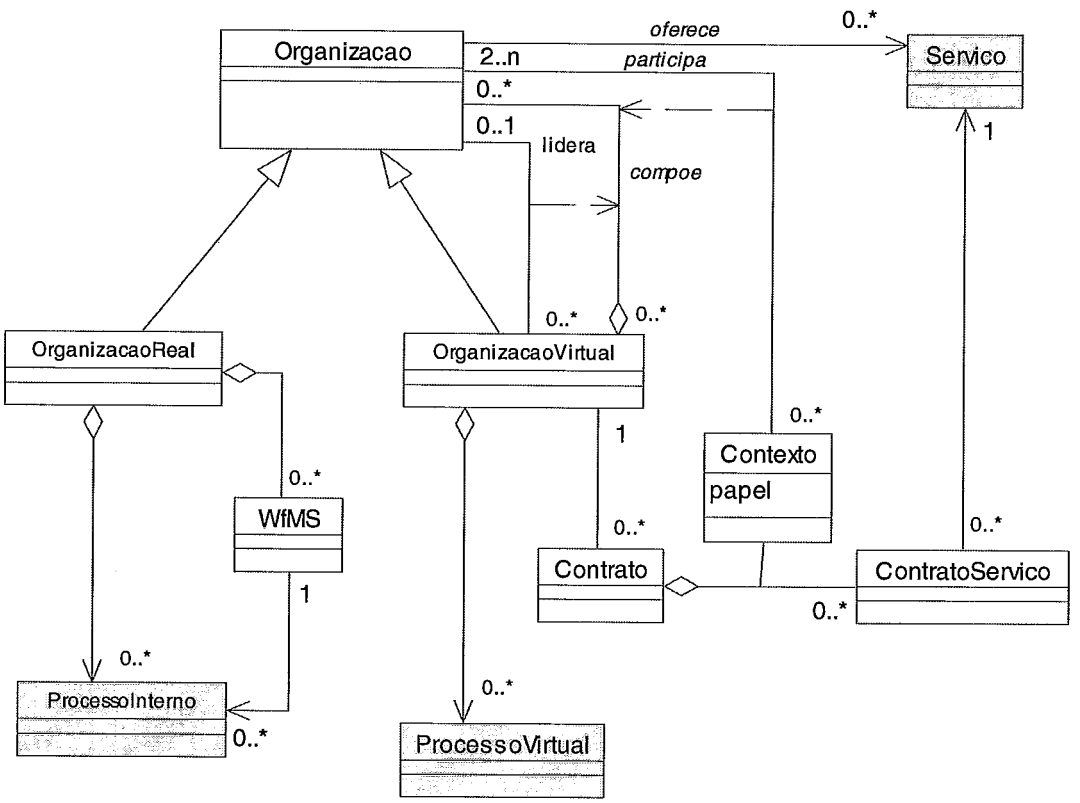

Figura 5.7 – Categoria Organização

Descrição sumária das classes:

No modelo de dados proposto foram considerados dois tipos de organização (classe Organizacao): organizações reais (classe OrganizacaoReal), que são organizações tradicionais, com estabelecimento físico, como nós as conhecemos no mundo "real"; e organizações virtuais (classe OrganizacaoVirtual), que são organizações formadas virtualmente a partir da composição de outras organizações, reais ou até mesmo virtuais, com o intuito de oferecer um serviço ou produto diferenciado em atenção a uma demanda do mercado. É importante ressaltar que o modelo proposto atende tanto ao modelo de empresas virtuais criadas *a priori* quanto ao modelo de empresas virtuais criadas dinamicamente.

Uma organização virtual pode ser liderada por uma das organizações que a constituem, caso ela seja gerenciada segundo uma administração centralizada. Nesse tipo de organização uma empresa predominantemente domina e coordena os processos virtuais.

Uma organização real possui processos internos (classe ProcessoInterno da categoria Processo). Possui também sistemas gerenciadores de *workflow* (WfMS), onde são definidos e encenados os processos internos.

A organização virtual possui um ou mais processos (relacionamento com a classe ProcessoVirtual da categoria Processo), que são os processos que atenderão à demanda que levou a empresa virtual a ser formada. A organização virtual é estabelecida a partir de um Contrato firmado entre as organizações que a compõe e esse contrato especifica como os processos virtuais estarão sendo executados no âmbito da empresa virtual.

Uma parte importante do contrato diz respeito aos serviços que serão utilizados pela organização virtual. Um contrato de serviço (classe ContratoServico), relacionado ao contrato geral, define os critérios de uso do serviço, referentes à qualidade do serviço, seu desempenho, tempo em que estará disponível ou quantidade de vezes que poderá ser executado, entre outros.

Cada organização assume um papel no Contexto de um contrato em relação a um contrato de serviço. O conjunto de organizações que participam de um Contexto deve ser o mesmo conjunto que compõe a organização virtual, como expressa o relacionamento de dependência entre as associações participa e compõe. No caso do modelo de empresas dinâmicas, haverá nitidamente uma organização contratante e uma organização contratada,

ou seja, uma empresa que utilizará o serviço fornecido por outra. No modelo de empresa formada *a priori*, o processo virtual é mais distribuído entre as organizações e, portanto, não há distinção tão precisa em relação ao uso do serviço, sendo possível identificar de maneira clara somente quem fornece o serviço. Logo, pode-se dizer que cada organização assume ou um papel de fornecedora ou neutro em relação ao serviço, nesse caso.

5.2. Arquitetura

Essa seção apresenta a arquitetura adotada para o desenvolvimento da primeira versão do sistema. Uma visão geral será apresentada para mostrar como estão organizados seus componentes. Em seguida os componentes da arquitetura serão descritos mais detalhadamente.

5.2.1 Visão Geral

Para a primeira versão do sistema foi adotada uma arquitetura distribuída, baseada no paradigma de chamada remota de procedimento (RPC). Assim, podem-se utilizar os benefícios da tecnologia de *web services*, como o acesso independente de plataforma via *web*. A Figura 5.8 ilustra essa arquitetura.

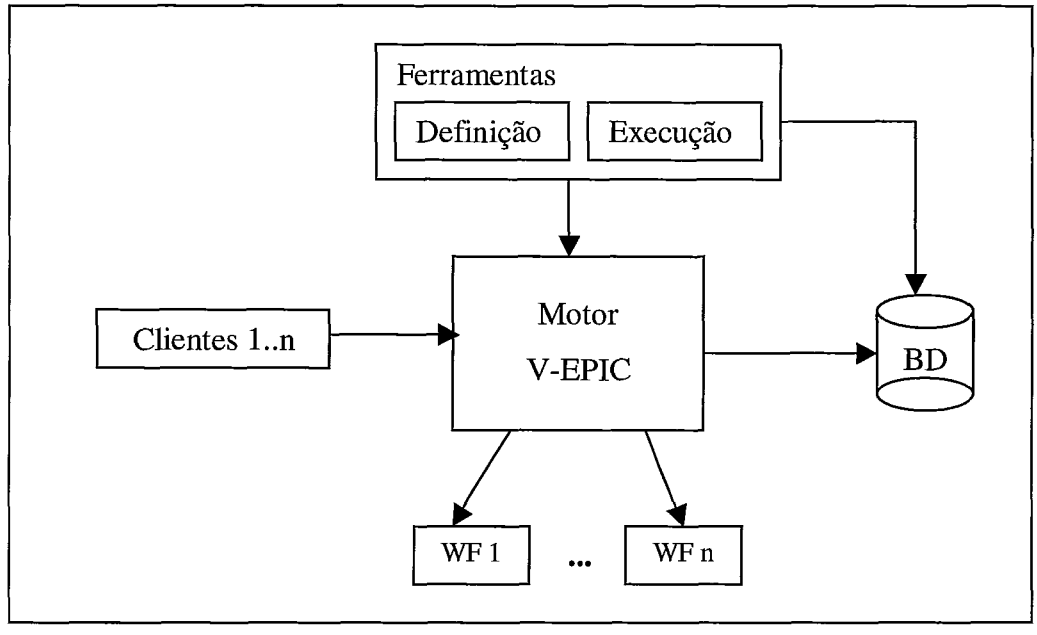

Figura 5.8 – Arquitetura do Virtual-EPIC – Visão Geral

O Motor Virtual-EPIC (Virtual-Enterprise Process Integration Coordinator) tem como objetivos principais a oferta de serviços a outras empresas, ou clientes do sistema, e a encenação de processos virtuais. Os clientes podem monitorar e controlar serviços. A comunicação entre os clientes e o Motor é feita através de mensagens XML sobre o protocolo de transporte SOAP (W3C-SOAP, 2000). O Motor, por sua vez, pode ser cliente de outros motores em outras empresas, caso haja processos virtuais sendo executados nele que utilizem serviços externos.

O ambiente se comunica com sistemas de gerenciamento de *workflow* comerciais, que provêem os processos que serão utilizados como implementação dos serviços oferecidos. A comunicação entre o Motor e os WfMSs comerciais é feita através de APIs específicas para cada produto, através de adaptadores, como será visto na próxima Sub-Seção, ou através da interface WAPI, padrão criado pela WfMC.

Os dados utilizados pelo ambiente serão armazenados em um sistema gerenciador de banco de dados comercial, segundo o esquema mostrado na Seção 5.1. Para essa versão da arquitetura foi escolhido utilizar um SGBD relacional, por ser esse tipo de SGBD mais utilizado no mercado e por haver para ele um padrão maduro de conexão genérica, o ODBC (ODBC, 1999). Na verdade, toda a comunicação com o SGBD será feita através de interface JDBC (JDBC, 2001), pois a programação foi feita em Java, como será visto em mais detalhes no Capítulo 7, e assim qualquer SGBDR comercial poderá ser adotado, desde que possua um *driver* JDBC ou um simples *driver* ODBC.

Algumas ferramentas são utilizadas para permitir que usuários acessem e manipulem as informações usadas pelo ambiente. Elas foram divididas em dois grupos: ferramentas de definição, responsáveis pela criação, definição e estruturação de informações relativas a tipos, serviços e processos virtuais; e ferramentas de execução, utilizadas para tratamento de informações e monitoramento de atividades durante a execução de instâncias de serviços e processos virtuais. Esses dois grupos de ferramentas atendem a três categorias de usuários: as ferramentas de definição são utilizadas por projetistas, enquanto as ferramentas de execução são utilizadas por participantes e monitores dos serviços. Essas categorias de usuários, obviamente, podem conter interseção, isto é, uma mesma pessoa pode participar de só uma das categorias, de duas delas ou até mesmo das três ao mesmo tempo.

5.2.2. Motor V-EPIC

O Motor é o núcleo da arquitetura e o seu componente mais importante. Como dito anteriormente, as funções principais do motor são oferecer serviços a outras empresas e encenar processos virtuais. Outra função importante exercida por ele é a gerência de organizações e contratos para estabelecimento de oferta/utilização de serviços nos processos virtuais.

O Motor é dividido em seis componentes: Gerenciador Central, Gerenciador de Contratos, Gerenciador de Processos Virtuais, Gerenciador de Serviços, WfMS Abstrato e Adaptadores de WfMSs, como pode ser visto na Figura 5.9.

Dentro do Motor é realizado o trabalho de mapeamento entre os serviços oferecidos e os processos internos que rodam nos WfMSs da empresa. Dentro dele há também o gerenciamento de processos virtuais, que por sua vez podem utilizar serviços tanto internos quanto de outras empresas, através de comunicação com os motores dos sistemas dessas empresas.

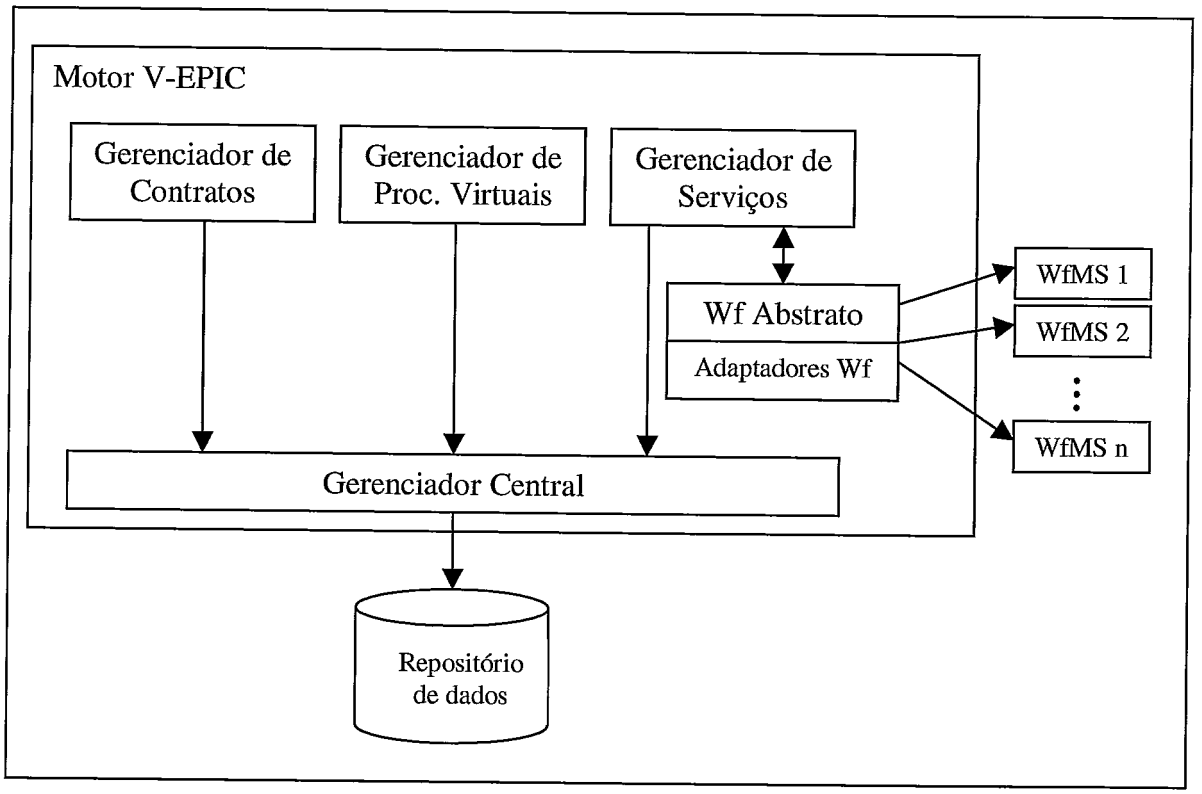

Figura 5.9 – Arquitetura do V-EPIC – Visão do Motor

Gerenciador Central

O Gerenciador Central assume o papel de coordenador dos demais gerenciadores. Quando o ambiente é iniciado, uma instância desse gerenciador é criada, que por sua vez cria instâncias de todos os demais gerenciadores. A comunicação entre os clientes, assim como as ferramentas, e o motor é feita por intermédio do Gerenciador Central.

É função do Gerenciador Central validar o acesso dos clientes e autorizar (ou não) suas ações, a fim de garantir a segurança do sistema. Da mesma forma, é verificada a identidade (*login* e senha) dos usuários das ferramentas, bem como suas permissões de acesso aos dados.

O acesso aos dados armazenados no repositório é também de responsabilidade desse gerenciador. Ele possui a identificação/localização dos documentos armazenados e a passa para os demais gerenciadores, quando esses o requisitam. A partir de então a base é acessada diretamente pelos gerenciadores.

Gerenciador de Contratos

O Gerenciador de Contratos tem a responsabilidade de permitir a manipulação de informações relativas à criação de organizações virtuais, seu estabelecimento através de contratos e garantir a correta definição e execução dos contratos de serviços. Esse gerenciador pode ser visto como a parte "burocrática" do ambiente, por tratar de informações declarativas formais, isto é, que podem vir a ser utilizada legalmente pelas partes envolvidas.

As organizações clientes do ambiente podem consultar os contratos registrados pela empresas, assim como alterar algumas informações relativas a seu cadastro, se possuir autorização para tal ação. Para isso, o Gerenciador de Contratos foi dividido em duas partes, inspirado na arquitetura de registros criada pela norma ebXML (EBXML-TA, 2001): uma interface, que permite o acesso de outras organizações às informações armazenadas e sua implementação real. Esse gerenciador pode ainda se comunicar com outros serviços de registro para acessar suas informações, tais como UDDI (UDDI, 2000), CORBA *Trading Service* (OMG-TS, 1996) e ebXML R*egistry* (EBXML-TA, 2001).

Gerenciador de Processos Virtuais

O objetivo do Gerenciador de Processos Virtuais é encenar, quer dizer, coordenar a execução das instâncias de processos virtuais que estejam em andamento no ambiente. Para cada atividade de uma instância de processo virtual, esse gerenciador verifica como foi declarada a sua implementação e instancia um serviço de acordo, utilizando para tanto as funcionalidades do Gerenciador de Serviços.

Quando uma atividade termina ou muda de estado, o Gerenciador de Processos Virtuais verifica qual é a próxima atividade (ou atividades) de acordo com as regras de transições definidas na definição do processo virtual e então inicia a sua execução.

Portanto, esse gerenciador acessa e manipula objetos relacionados tanto a definições de processos virtuais e seus respectivos dados quanto a instâncias de processos virtuais. Ele registra também referências aos serviços (internos e externos) utilizados nas instâncias virtuais.

É importante lembrar que um serviço pode ter como implementação um processo virtual, quando o serviço é composto de outros serviços. Nesse caso o Gerenciador de Serviços é responsável pela coordenação da execução da implementação do serviço.

Gerenciador de Serviços

O Gerenciador de Serviços tem como principal função realizar o trabalho de mapeamento entre os serviços oferecidos pela organização e os processos internos que rodam nos seus WfMSs. Esse gerenciador fica permanentemente monitorando as instâncias de processos que foram encapsuladas em serviços, a fim de permitir o monitoramento de seus estados pelos clientes.

O Gerenciador de Serviços oferece também a possibilidade de controle sobre os serviços aos clientes do ambiente. A partir de comandos de controle recebidos pelo motor esse gerenciador invoca uma mudança de estados no serviço correspondente e, consequentemente, uma ação no processo é realizada para, por exemplo, fazê-lo mudar de atividade, segundo a implementação da respectiva operação no modelo do serviço.

Cada vez que acontece algo relevante a um serviço, como por exemplo, uma mudança de estado, uma exceção ou erro ou o fim da execução do serviço, uma notificação é enviada a todos os clientes que tenham registrado interesse nesse tipo de evento. A

configuração de notificações relativas a serviços é feita através de uma das ferramentas acessórias ao ambiente (ver Seção 5.2.3).

WfMS Abstrato e Adaptadores de WfMSs

Como foi visto no Capítulo 4, um importante requisito da solução é permitir que possam ser usados como implementações de serviços processos que estão em operação nas organizações, definidos e encenados em sistemas de gerenciamento de *workflow* comerciais. Para atender a esse requisito, o ambiente tem que conseguir "enxergar" esses processos, estejam eles definidos em qualquer WfMS comercial disponível na organização.

A forma adotada para solucionar essa questão foi utilizar uma "máquina virtual" de *workflow*, isto é, um WfMS abstrato, neutro, que seja independente de WfMSs específicos. Todos os outros gerenciadores se comunicam com esse WfMS neutro, sem se preocupar com detalhes específicos de produtos comerciais. O WfMS abstrato, então, fica encarregado de transformar o pedido de informação ou a alteração de dados enviados numa chamada ao sistema real de *workflow*. O acesso ao sistema comercial é feito por intermédio de um adaptador, que "traduz" o comando genérico para a API do WfMS comercial.

Os comandos que o WfMS abstrato reconhece foram baseados e estão de acordo na interface 2 da WfMC (WfMC-TC-1009, 1998). Um adaptador genérico, compatível com a interface 2 é instanciado por *default*, permitindo assim que sistemas comerciais que obedeçam a esse padrão sejam usados pelo ambiente de forma direta e transparente.

No entanto, para cada WfMS comercial não-compatível com a WfMC quanto à interface 2 deverá ser criado um adaptador a partir de sua API ou através do meio mais propício oferecido pelo sistema. A Figura 5.10 mostra alguns exemplos de adaptadores integrados ao *Workflow* Abstrato para a comunicação com os respectivos sistemas comerciais.

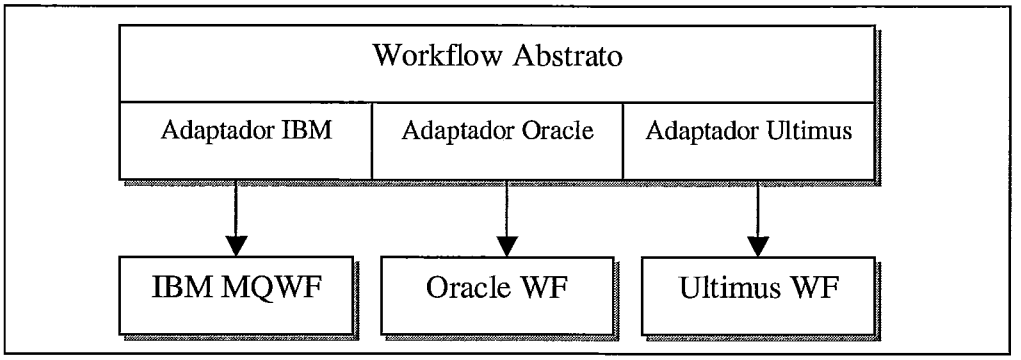

Figura 5.10 – Exemplo de Adaptadores para o *Workflow* Abstrato

5.2.3. Ferramentas Acessórias

As ferramentas acessórias ao ambiente visam auxiliar o trabalho das três categorias de usuários que o utilizam: os projetistas, os monitores e os participantes, conforme foi dito anteriormente. Logo, as ferramentas foram definidas tendo em mente as funções exercidas por essas pessoas.

A função do projetista é criar os tipos de esquemas de estados que servirão de base para a definição de serviços; criar os serviços em si, contemplando a definição de interfaces e sua implementação; cadastrar organizações e gerenciar a criação de organizações virtuais e informações relacionadas a contratos e, por fim, definir processos virtuais e suas atividades.

A função do participante é escolher serviços dinamicamente no ato de criação de instâncias de processos virtuais, entre os serviços disponíveis para cada atividade do processo, de acordo com critérios de qualidade, preço, etc. dos serviços.

A função do monitor é controlar e monitorar serviços externos, isto é, serviços que estarão sendo executados em outras organizações. O monitor deverá ter acesso aos estados dos serviços e terá permissão para invocar medidas de controles dos serviços, como, por exemplo, alterar o estado em que o serviço se encontra ou terminá-lo. O monitor poderá receber notificações de alterações de estados e outros eventos relevantes de serviços, se assim o desejar.

A seguir é mostrada uma lista de ferramentas criadas para atender a essas funções, divididas por categorias de usuários:

Projetista:

- Ferramenta de definição de organizações virtuais e gerenciamento de contratos;
- Ferramenta de definição de tipos de esquemas de estados;
- Ferramenta de especificação de processos virtuais
- Ferramenta de especificação de serviços
- Ferramenta de especificação de *awareness* (para monitores)

Participante:

- Ferramenta de escolha dinâmica de serviço

Monitor:

- Ferramenta de visualização de estados de serviços e controle sobre os mesmos

No Capítulo 6 serão apresentados os projetos de algumas dessas ferramentas, assim como suas características técnicas em geral.

5.2.4. Arquiteturas Alternativas

A arquitetura proposta, apesar de simples, atende bem aos requisitos identificados para integração de processos entre empresas. Contudo, esse modelo de arquitetura é mais adequado ao modelo de empresa virtual onde há "terceirização" de serviços, ou seja, no modelo em que uma empresa aloca a outras empresas partes do seu processo, pois há aí uma forte coordenação centralizada, de acordo com a arquitetura que também possui características centralizadoras.

Para o modelo em que o processo virtual é mais homogeneamente distribuído, uma arquitetura onde seus componentes estejam mais distribuídos poderia vir a atender melhor à empresa virtual. Nesse caso, algumas funcionalidades contidas no motor do ambiente seriam retiradas do ambiente e colocadas à parte do ambiente, passando a atuar de maneira bastante autônoma em relação ao ambiente e independente de qualquer empresa específica. Esses componentes poderiam ser postos numa locação neutra pertencente à empresa virtual, isto é, fora do domínio das organizações que a constituem, ou dentro de uma das organizações, porém administrado separadamente.

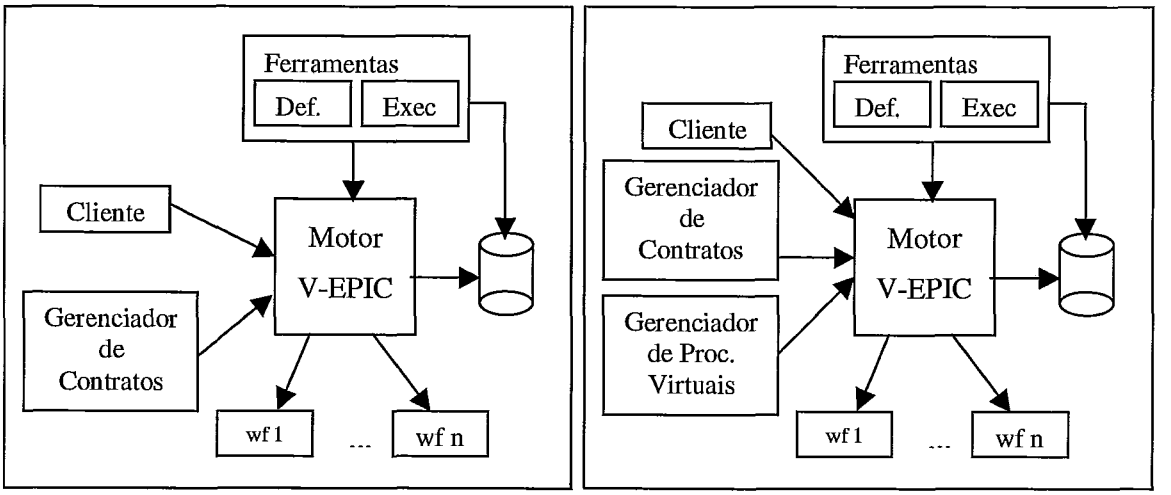

Figura 5.11 – Arquiteturas Alternativas – a) Gerenciador de Contratos à parte – b) Gerenciadores de Contrato e de Processos Virtuais à parte

A Figura 5.11 apresenta duas possibilidades de arquiteturas alternativas. Na primeira alternativa, o Gerenciador de Contratos é separado ambiente. Na segunda, também o Gerenciador de Processos virtuais é separado.

Na alternativa ilustrada na Figura 5.11 a), o Gerenciador de Contratos é retirado do motor do ambiente e posto à parte. Isso permite que definições de organizações físicas e virtuais sejam administradas de forma neutra em relação a qualquer ambiente, ou seja, sem a intervenção de nenhuma organização específica. Essa arquitetura está de acordo com as novas arquiteturas baseadas em Web Services, pois o gerenciador de contratos seria, de certa forma, semelhante aos registros públicos de serviços, como, por exemplo, os repositórios UDDI e registros ebXML.

A Figura 5.11 b) apresenta outra alternativa, ainda mais distribuída. Nessa, o gerenciador de processos virtuais também se encontra em ambiente neutro, o que possibilita a definição e administração desses processos por entidades neutras.

5.3. Conclusões Preliminares

Esse capítulo apresentou dois elementos fundamentais para a construção do ambiente: o modelo de dados que será utilizado como alicerce da estruturação de informações; e a arquitetura, que servirá como infra-estrutura de componentes para a sua construção, como será visto no Capítulo 6.

A elaboração do modelo de dados procurou, sempre que possível, manter compatibilidade com os conceitos e definições da WfMC e OMG. Do modelo de dados resultará o esquema a ser armazenado no SGBD.

6 - Projeto da Infra-Estrutura

O objetivo desse capítulo é apresentar o projeto dos principais componentes previstos na arquitetura e modelo de dados, mostrados no Capítulo 5. O projeto leva em consideração, sempre que possível, os requisitos, conceitos e idéias de soluções apresentados no Capítulo 4, de forma a prover um meio de realizá-los através da implementação.

O projeto elaborado procura utilizar as melhores práticas de projeto de software, no intuito de fornecer uma infra-estrutura extensível, flexível e, consequentemente, facilmente manutenível. Como artifícios para atingir esse fim, em diversas ocasiões foram utilizados *design patterns* (GAMMA et al., 1995, COPLIEN, 1992, DAVIS, 1999, COOPER, 1998). Um design pattern captura a estrutura, a compreensão e os principais aspectos de uma solução para um problema recorrente de projeto de software dentro de um certo contexto e que pode ser reutilizada inúmeras vezes. Logo, percebe-se que o objetivo maior de um *design pattern* é tornar mais fácil e bem sucedida a reutilização de bons projetos de software. A maioria dos *design patterns* utilizados nesse trabalho podem ser encontrados no catálogo definido por GAMMA et al. (1995). Muitos deles serão descritos no decorrer desse capítulo.

Será mostrado também o projeto de um *framework* para o desenvolvimento das ferramentas acessórias, tanto de definição quanto de execução. Foram contempladas as funcionalidades básicas de definição, a saber: definição de tipos, processos virtuais, serviços e contratos; e de execução: instanciação de processos virtuais.

Por último, será explicado como pode ser criada uma nova aplicação utilizando-se a infra-estrutura. Um roteiro de etapas necessário ao desenvolvimento será mostrado, com ênfase no mecanismo de especialização que deve ser adotado para estender a infra-estrutura.

A Seção 6.1 mostra o projeto dos componentes básicos da arquitetura. Na Seção 6.2, acha-se a descrição do projeto do *framework* de ferramentas acessórias. Por fim, a Seção 6.3 explica como pode ser feito o desenvolvimento de novas aplicações.

6.1. Projeto dos Componentes Básicos da Arquitetura

Esta Seção apresentará o projeto dos principais componentes da arquitetura, evidenciando as questões surgidas no seu desenvolvimento e as decisões de projeto tomadas para sua elucidação. Apresentar-se-á, inicialmente, o projeto da categoria Esquema de Estados, que representa os tipos básicos utilizados por alguns dos demais gerenciadores e componentes em geral. Em seguida, serão discutidos todos os gerenciadores e componentes apontados na arquitetura: Gerenciador de Contratos, WfMS Abstrato, Gerenciador de Serviços, Gerenciador de Processos Virtuais e, finalmente, o Gerenciador Central, coordenador de todos os anteriores.

6.1.1. Categoria Esquema de Estados

A categoria Esquema de Estados (EsquemaEstados) oferece a toda a infra-estrutura uma forma de representação e utilização do tipo básico (classe TipoEsquemaEstados). Antes de apresentar essa categoria, contudo, apresentaremos outra categoria, utilizada por ela, chamada Expressão.

- Categoria Expressão

A categoria Expressão é uma categoria típica de projeto e provê meios para a representação e interpretação de expressões *booleanas* em geral. É um elemento importantíssimo do projeto, pois permite a definição e posterior monitoramento de expressões relacionadas às transições de estados, no mapeamento dos serviços aos processos internos. O projeto desta categoria reflete o *design pattern Interpreter*, que pode ser observado na Figura 6.1

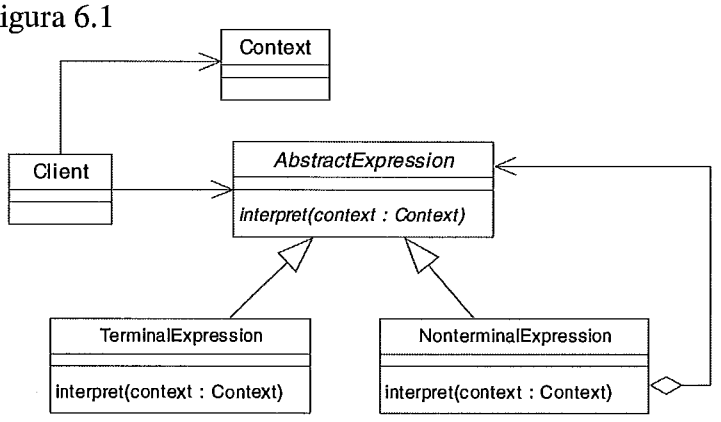

Figura 6.1 – *Design Pattern Interpreter* (baseado em GAMMA et al., 1995)

O *design pattern Interpreter* é aplicável em situações onde se deseja representar linguagens possíveis de serem expressas em gramáticas simples. A classe Abstract Expression é uma classe abstrata que define o método interpret. Os elementos da linguagem são representados por expressões não terminais (NonTerminalExpression), isto é, que possuem em sua expressão outras expressões e por expressões terminais (TerminalExpression). Como exemplo de expressões terminais podemos citar os literais utilizados numa linguagem de programação. A classe Context representa um contexto de representação e posterior interpretação. Nessa classe, por exemplo, são feitos os assinalamentos de variáveis a expressões. Na hora da interpretação, toda vez que uma variável é encontrada o interpretador recorre à classe Context para interpretar seu real valor.

A gramática definida para a representação de expressões no projeto, descrita em notação BNF (Forma de Backus-Naur) (NAUR, 1963), pode ser observada a seguir:

BooleanExp ::= VariableExp | Constant | OrExp | AndExp | NotExp | EqualExp | EqualInt | NotEqualExp | NotEqualInt | IntExp
AndExp ::= BooleanExp 'and' BooleanExp
OrExp ::= BooleanExp 'or' BooleanExp
NotExp ::= 'not' BooleanExp
Constant ::= 'true' | 'false'
EqualExp ::= BooleanExp = BooleanExp
EqualIntExp ::= IntExp = IntExp
NotEqualExp ::= BooleanExp <> BooleanExp
NotEqualInt ::= IntExp <> IntExp
VariableExp ::= 'a' | 'b'| ... | 'x' | 'y' | 'z'
IntExp ::= IntVariableExp | ObjectState | StateConstant
StateConstant ::= 'pronto' | 'executando' | 'suspenso' | 'terminado' | 'completo'
ObjectState ::= objeto.getState()
IntVariableExp ::= 'a' | 'b'| ... | 'x' | 'y' | 'z'

A Figura 6.2. apresenta, de maneira simplificada, a definição dessa gramática através do *design pattern Interpreter* na categoria Expressão. Uma das vantagens da utilização deste tipo de projeto é a possibilidade de alterar e estender facilmente a linguagem definida, criando-se novas regras a partir de novas especializações. Isto é, para

adicionar-se uma nova regra à gramática adotada, basta criar uma nova especialização da classe BooleanExp.

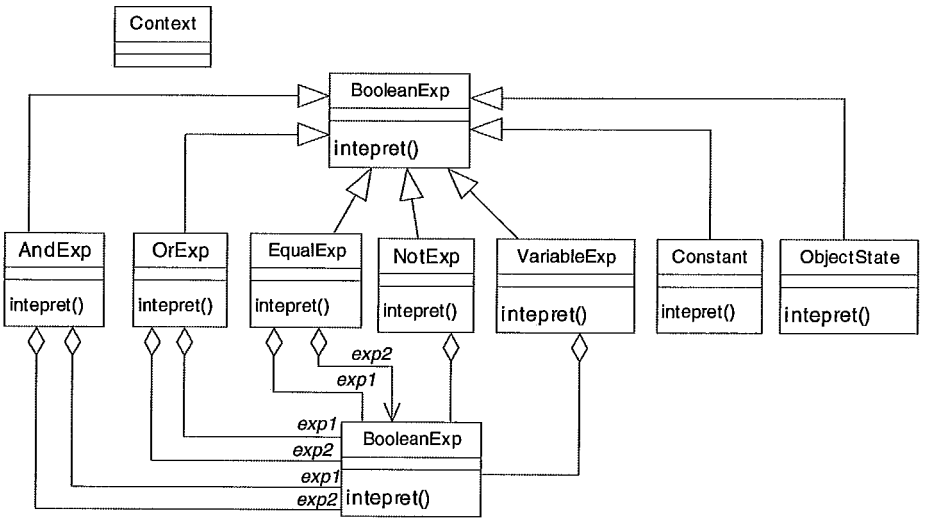

Figura 6.2 – Categoria Expressão

Essa gramática permite a representação de expressões aninhadas complexas. Por exemplo, pode-se representar a seguinte expressão:

expressao := (Ativ1.estado = 'completo' ou Ativ2.estado <> 'suspenso') e

(não (Ativ3.estado = x e Ativ4.estado <> y)))

Nesse contexto, as variáveis x e y foram assinaladas aos seguintes valores:

x = 'pronto', y = 'terminado'.

Na Figura 6.3, pode-se observar um diagrama de objetos correspondente à expressão dada como exemplo. Nota-se que a estrutura assemelha-se a uma árvore.

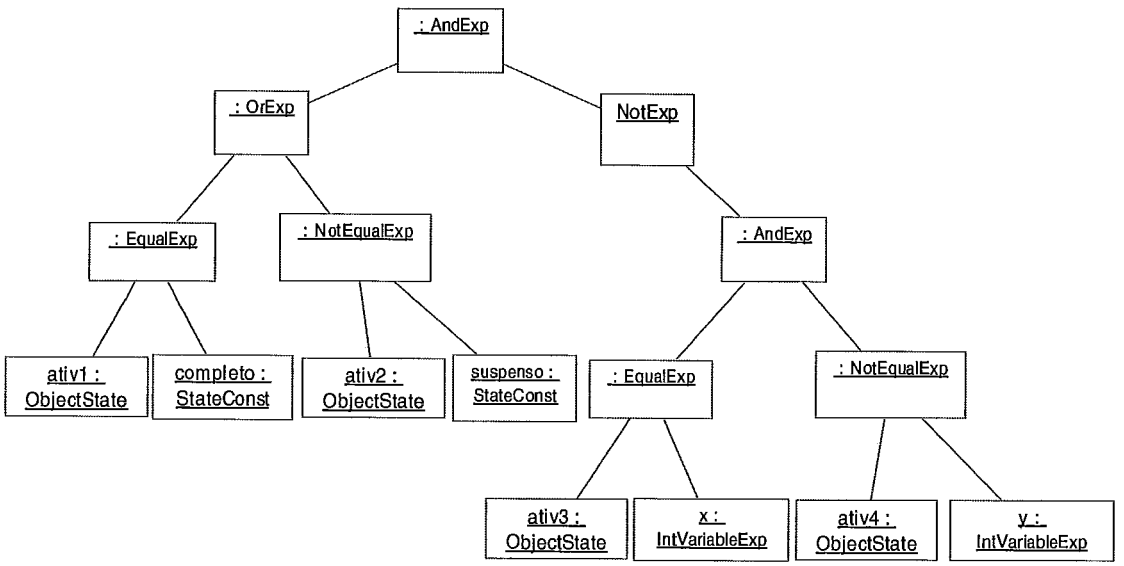

Figura 6.3 – Exemplo de Árvore de Objetos representando uma Expressão

Se para uma instância de processo, num determinado instante, o estado da Atividade 1 for igual a 'completo', o estado da Atividade 2 for diferente de 'suspenso', o estado da Atividade 3 for igual a 'pronto' e o estado da Atividade 4 for igual a 'terminado', a expressão será interpretada como verdadeira e uma transição de estado ocorrerá.

➢ Categoria Esquema de Estados

É na categoria Esquema de Estados (Figura 6.4) que acontece o mapeamento dos processos internos para a abstração de estados utilizada pelos serviços. Portanto, é de fundamental importância para o ambiente. Serão apresentadas, agora, as decisões de projeto referentes a essa categoria, mais especificamente, sobre a definição e uso de condições de transições de estados e o armazenamento e utilização de operações.

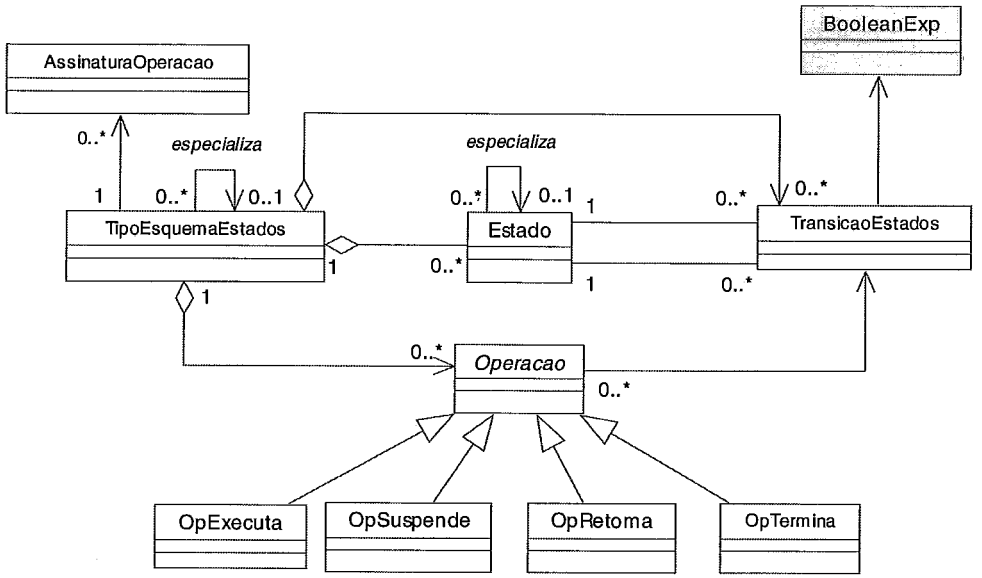

Figura 6.4 – Categoria Esquema de Estados (Projeto)

Como visto no Capítulo 5, uma transição de estados (classe TransicaoEstados) armazena uma condição que é constantemente verificada e, ao ser interpretada como verdadeira, desencadeia a transição. Essa condição é expressa na classe BooleanExp. A fim de armazenar-se uma árvore de objetos referente a uma expressão (como a árvore vista na Figura 6.3), utilizou-se o mecanismo de serialização de objetos da linguagem Java (CAMPIONE et al., 1999). Através deste mecanismo, objetos e toda sua rede de referências podem ser escritos e lidos em arquivos físicos.

Na fase de definição, indica-se somente a referência da localização do arquivo que contém o objeto serializado que representa a expressão relacionada à condição de transição. Ao utilizar-se pela primeira vez um tipo de esquema de estados (classe TipoEsquemaEstados), carrega-se na memória primária o objeto, que é associado à classe TransicaoEstados. Daí em diante se utiliza somente a referência em memória primária.

As operações (classe Operacao) têm como função permitir o controle sobre serviços. Ao se invocar uma operação, ocasiona-se uma transição de estados. Algumas operações "padrões" foram previamente definidas para o tipo básico WfMC (conforme explicado no Capítulo 5): OpExecuta, que causa uma transição para o estado genérico "Executando", OpSuspende, que ocasiona uma mudança para o estado "suspenso", OpRetoma, que faz o serviço voltar ao estado "Executando" após ter sido suspenso e, finalmente, OpTermina, que faz o serviço transitar para o estado "Terminado". Essas quatro classes básicas funcionam como um *framework* de onde podem ser criadas novas operações quando são definidos novos estados em novos tipos. Por exemplo, caso um tipo possua dois novos estados especializados do estado "executando", "executando serviço técnico" e "cobrando serviço", duas novas especializações de operações (da operação OpExecuta) podem ser definidas a fim de gerar transições para esses novos estados.

Os refinamentos de operações são feitos especializando-se as classes responsáveis pelas operações pré-definidas na linguagem de programação. Nessas novas classes, são programadas as atividades necessárias à realização da operação no nível do processo, utilizando o WfMS abstrato, isto é, sem ser necessário se ater a particularidades de produtos específicos e preocupando-se somente com o processo em si. No entanto, no instante de definição são indicadas somente as assinaturas das operações (classe AssinaturaOperacao), com os atributos "nome da operação", "lista de tipos de parâmetros", "tipo do objeto de retorno", "localização da classe" (em meio físico) e "estado de origem" e "estado de destino". No momento de execução, novos objetos são instanciados a partir dessas classes, segundo o seu tipo de operação, e faz-se a ligação ao objeto TransicaoEstados correspondente.

6.1.2. Categoria Gerenciador de Contratos

O Gerenciador de Contratos, mostrado na Figura 6.5, funciona como uma espécie de "arquivo geral". Entre suas funcionalidades estão o acesso de leitura e escrita aos documentos "burocráticos" (cadastros de organizações e contratos) utilizados pelo ambiente.

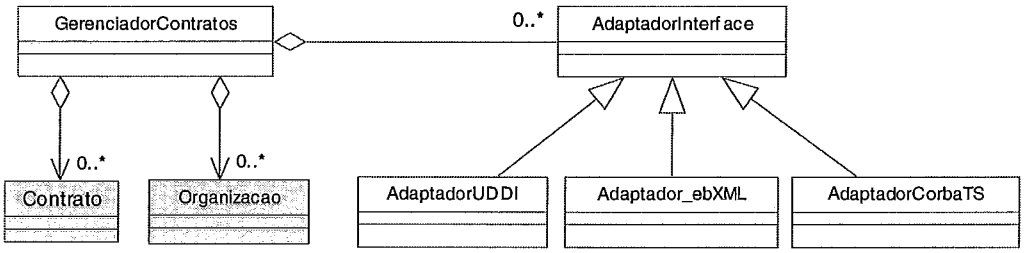

Figura 6.5 – Categoria Gerenciador de Contratos

A Figura 6.5 mostra as classes dessa categoria. A classe Gerenciador de Contratos (GerenciadorContratos) funciona como coordenador central dessa categoria. Com o intuito de facilitar e otimizar o acesso aos contratos e cadastros de organizações, listas de objetos dessas classes são armazenadas. Quando uma organização ou contrato é consultado/alterado pela primeira vez, são recuperados do banco. Depois, ficam armazenados de forma indexada no registro da classe GerenciadorContratos, permitindo assim um acesso mais rápido.

Como foi explicado no Capítulo 5, o Gerenciador de Contratos pode consultar outros mecanismos de registro de serviços, como UDDI, registro ebXML e CORBA Trading Service. O acesso a essas interfaces é feito através de adaptadores para esses registros (especializações da classe AdaptadorInterface), como pode ser visualizado na Figura 6.5. Para se adicionar novos tipos de acessos a novos mecanismos de registros, basta criar-se novas especializações da classe AdaptadorInterface.

6.1.3. *Workflow* Abstrato e Adaptadores

Será apresentado, agora, o projeto das classes responsáveis por permitir a utilização de um sistema gerenciador de *workflow* abstrato (WfMS abstrato) pelo Gerenciador de Serviços. Será mostrado também como podem ser criados adaptadores para os WfMSs específicos. O WfMS abstrato definido apresenta como interface um subconjunto das

funcionalidades presentes na interface 2, definida pela WfMC (WFNC-TC-1009, 1998), e na especificação para *workflow* criada pela OMG (OMG-WMF, 1998). Ou seja, as funcionalidades planejadas nessas classes são compatíveis com as da interface 2 da WfMC, quanto ao funcionamento esperado, métodos, parâmetros, etc. Foi escolhido um conjunto mínimo de funcionalidades de *workflow* que atendessem às necessidades da solução desenvolvida.

O conceito de projeto que está por trás dessas classes é o mesmo do *design pattern Bridge*. Seu objetivo é proporcionar a separação entre uma abstração e suas implementações. A implementação pode, até mesmo, ser escolhida em tempo de execução. Outra vantagem de seu uso é evitar que alterações feitas nas implementações afetem seus clientes. Ou seja, evita-se ter que reprogramar as classes clientes.

A Figura 6.6 ilustra o *design pattern Bridge*. Tudo que os métodos da classe Abstracation fazem é invocar operações da classe Implementor (na verdade, de uma de suas especializações). Contudo, os clientes (classe Client) só têm acesso ao método de Abstraction e, portanto, mudanças feitas nas classes de implementação não os afetam.

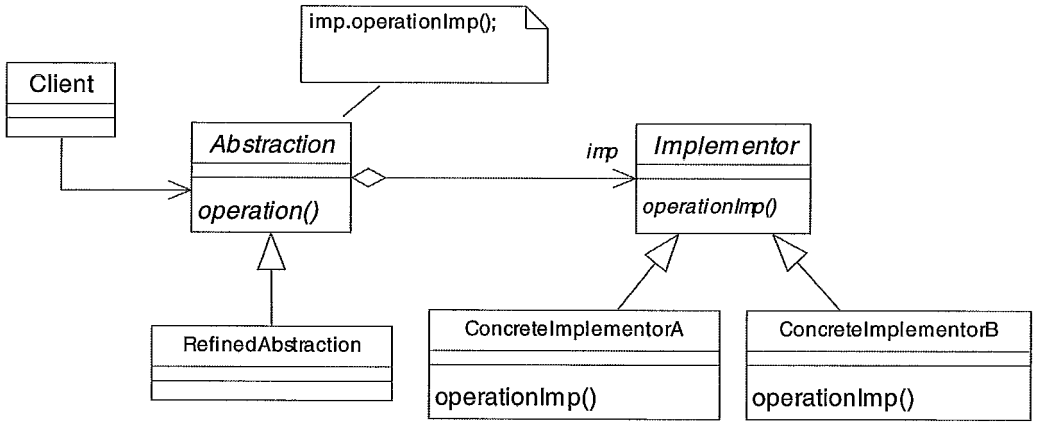

Figura 6.6 – *Design Pattern Bridge* (baseado em GAMMA et al., 1995)

A Figura 6.7 apresenta o projeto da categoria WfMSAbstrato, inspirado no *design pattern Bridge*. Nela, podem-se ver as classes WfMSAbstrato, InstanciaProcesso e InstanciaAtividade. De fato, foram utilizados três vezes o *pattern Bridge*, uma vez para cada uma dessas classes. Assim, criam-se abstrações para WfMSs, suas instâncias de processos e as instâncias de atividades relacionadas às instâncias de processo, respectivamente.

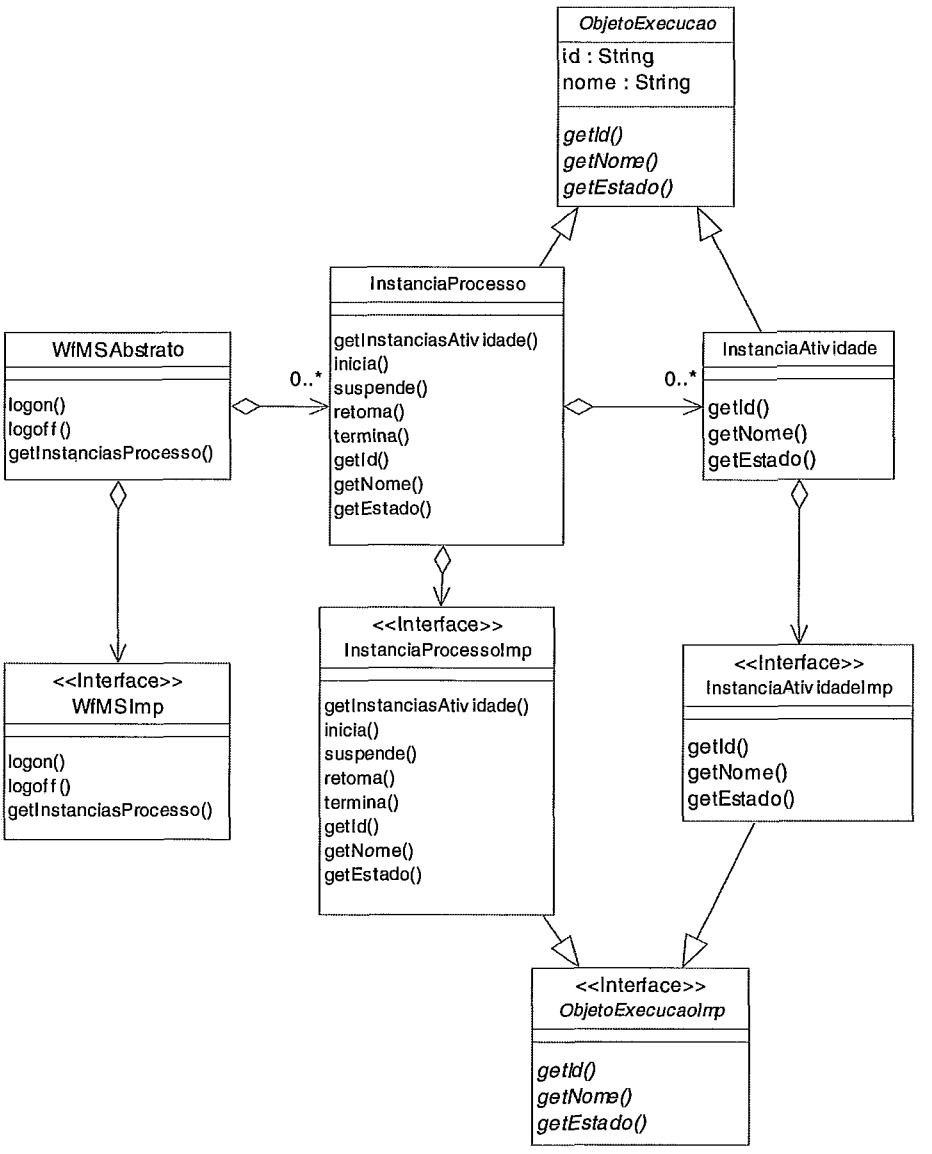

Figura 6.7 – Classes referentes ao WfMS Abstrato

A classe WfMSAbstrato oferece funcionalidades básicas de acesso a um sistema gerenciador de *workflow*, como conexão e desconexão, uma forma de se obter as instâncias de processos definidas e a possibilidade de instanciação de um processo.

As classes InstanciaProcesso e InstanciaAtividade possuem como generalização a classe Objeto de Execução (classe ObjetoExecucao), à semelhança da estrutura de classes definida pela OMG *Workflow Management Facility*. Os atributos presentes em ObjetoExecucao são nome, que identifica a instância de uma maneira compreensível através da leitura humana e id, que representa um identificador do objeto (na forma de

String, número ou qualquer outra). Os métodos dessa classe permitem obter o seu identificador (atributo id), seu nome e, o mais importante, o estado da instância.

A classe Instância de Processo implementa os métodos definidos em Objeto de Execução e define quatro novos métodos, para iniciar a execução da instância, suspendê-la, retomá-la após a suspensão e terminá-la/abortá-la, além de um método para se obter as instâncias de atividades relacionadas à instância de processos representada por ela. A classe Instância de Atividade só implementa os métodos definidos em sua classe pai, sem adicionar novos métodos.

As interfaces WfMSImp, InstanciaProcessoImp e InstanciaAtividadeImp equivalem, cada uma delas, à classe Implementor do *design pattern Bridge*, mostrada na Figura 6.6. As interfaces InstanciaProcessoImp e InstanciaAtividadeImp especializam uma interface genérica chamada ObjetoExecucaoImp, com o mesmo sentido da hierarquia definida na abstração.

Por motivos de otimização de memória, a classe WfMSAbstrato armazena somente o identificador das instâncias de processos que possui, postergando a referência verdadeira para objetos da classe InstanciaProcesso quando forem solicitadas. Assim, evita-se armazenar-se em memória primária todas as instâncias de processos em execução num WfMS. Esse é um dos objetivos do padrão de projeto *Proxy*, que pode ser visualizado na Figura 6.8.

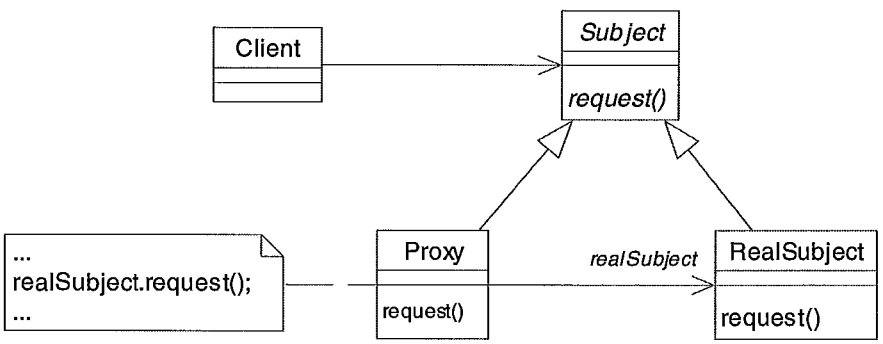

Figura 6.8 – *Design Pattern Proxy* (baseado em GAMMA et al., 1995)

Ambas as classes Proxy e RealSubject implementam uma mesma interface, definida pela classe abstrata Subject, quer dizer, oferecem os mesmos serviços. O objeto Proxy é utilizado até que seja realmente necessário o uso do objeto real (classe RealSubject), evitando-se assim o custo de manter-se em memória um objeto dispendioso. Esse tipo de

uso do padrão *Proxy* é chamado *Virtual Proxy*. Outros usos desse *design pattern* foram adotados, como será descrito adiante.

De modo a se utilizar produtos de gerenciamento de *workflow* específicos, deve-se especializar as interfaces WfMSImp, InstanciaProcessoImp e InstanciaAtividadeImp, sendo essas novas especializações equivalentes à classe ConcreteImplementor mostrada na Figura 6.6. Essas novas interfaces são criadas numa categoria à parte, sendo uma para cada nova implementação.

A Figura 6.9 apresenta um exemplo de categoria criada para um produto específico de gerenciamento de *workflow*, com suas classes e seus respectivos relacionamentos com as classes da categoria WfMSAbstrato. Nesta figura, torna-se claro que para criar-se um novo adaptador de WfMS, basta criar três classes, implementadoras das interfaces WfMSImp, InstanciaProcessoImp e InstanciaAtividadeImp.

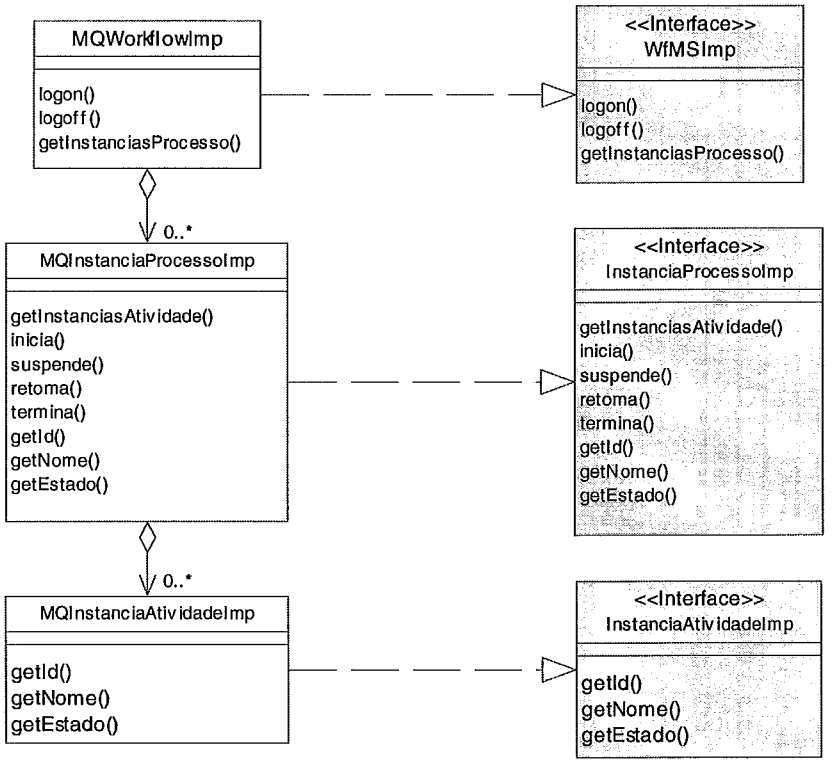

Figura 6.9 – Categoria MQWFImp: Exemplo de Categoria criada para a Implementação de um determinado WfMS

Resta ainda a seguinte questão referente ao projeto do WfMS Abstrato: como escolher a implementação correta, isto é, o WfMS comercial específico, durante a execução? A solução a essa questão está no *design pattern Factory Method*. GAMMA et

al., 1995 descrevem a seguinte aplicabilidade desse padrão: quando uma classe não pode antecipar a classe de objetos que deve criar.

A Figura 6.10 ilustra o padrão de projeto *Factory Method.* A classe abstrata Creator possui um método abstrato chamado factoryMethod, que retorna uma instância da classe genérica Product. A classe ConcreteCreator, que herda de Creator, implementa esse método para retornar um produto concreto (classe ConcreteProduct). É necessário criar novas especializações de Creator para cada novo produto criado.

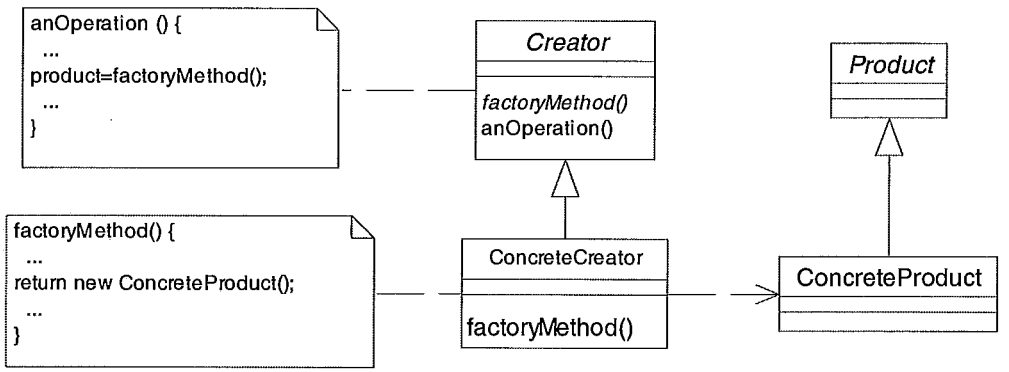

Figura 6.10 – *Design Pattern Factory Method* (baseado em GAMMA et al., 1995)

No projeto do ambiente, foi utilizada uma variação desse padrão, chamada *Parameterized Factory Method.* Sua estrutura, aplicada ao projeto desenvolvido, pode ser vista na Figura 6.11. A classe Fábrica WfMS (classe FabricaWfMS), equivalente à classe Creator, possui um método de criação (criarWfMS) que recebe como parâmetro o identificador do WfMS e retorna uma instância da interface WfMSImp apropriada.

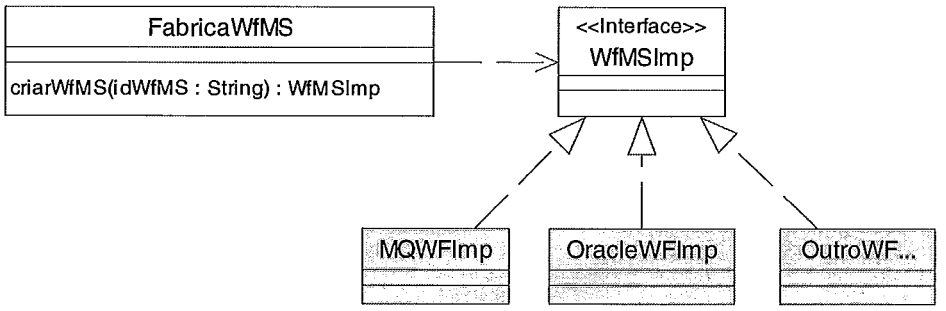

Figura 6.11 – Fábrica de WfMS (Categoria WfMS Abstrato)

Na Figura 6.11 podem-se observar ainda algumas especializações de WfMSImp, para diversos produtos de *workflow.* A classe FabricaWfMS, quando é criada, permite a criação de um conjunto inicial de produtos pré-estabelecidos. Ao adicionarem-se novos

adaptadores de WfMS, torna-se preciso especializar a classe FabricaWfMS para contemplar os novos produtos.

6.1.4. Categoria Gerenciador de Serviços

O objetivo da categoria Geenciador de Serviços (Figura 6.12) é permitir o monitoramento de instâncias de serviços, através do monitoramento das instâncias de processos mapeadas por eles, a fim de fornecer *feedback* sobre sua execução a clientes.

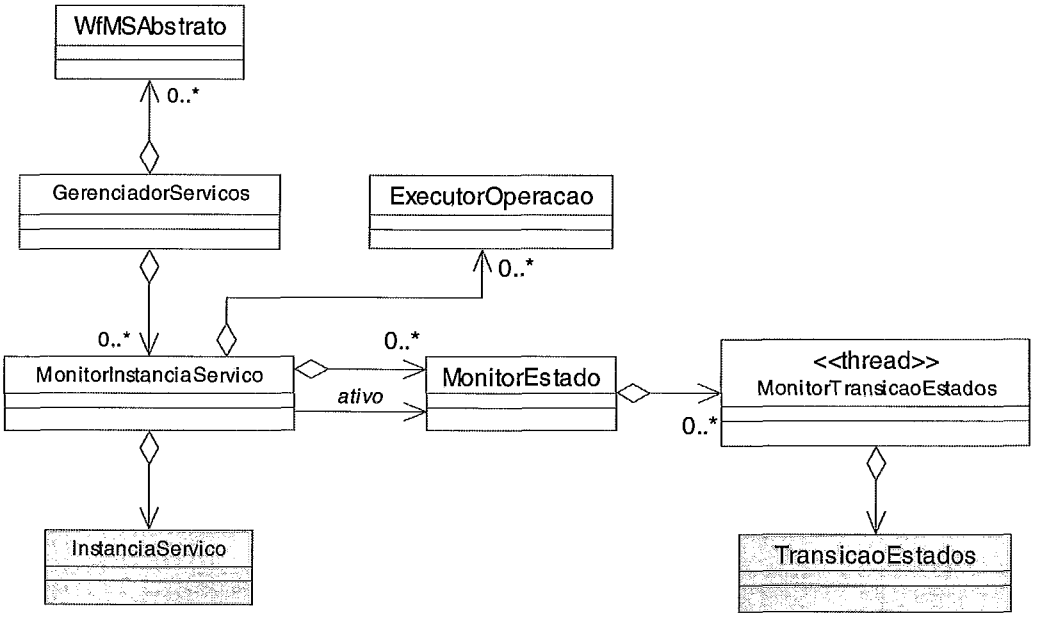

Figura 6.12 – Categoria Gerenciador de Serviços

A classe Gerenciador de Serviços funciona como um coordenador geral para as funcionalidades providas por sua categoria. Quando é solicitada a criação de uma instância de serviço, é criada também a instância de processo correspondente, por meio do WfMS Abstrato (WfMSAbstrato). Esse evento é registrado na classe InstanciaServico. A partir desse momento, a instância de serviço será permanentemente monitorada, a fim de detectarem-se mudanças de estados, até o fim de sua execução. Clientes poderão consultar instâncias de serviços para obter informações sobre seu estado.

O Gerenciador de Serviços possui diversos objetos da classe MonitorInstanciaServico, um para cada instância de serviço em execução. Por sua vez, essa classe possui diversas instâncias da classe ExecutorOperacao, uma para cada operação

definida no tipo que esse serviço implementa (definido no relacionamento de Servico com TipoEsquemaEstados).

Uma instância de serviço possui diversos objetos da classe MonitorEstado, um para cada Estado pelo qual o serviço pode passar. No entanto, somente um estado pode estar ativo a cada instante, isto é, uma instância de serviço não pode estar em dois estados ao mesmo tempo, pois isso seria uma inconsistência.

Cada transição de estados possível de ser executada em cada estado é registrada na classe Monitor de Transição de Estados (classe MonitorTransicaoEstados). Cada instância dessa classe é executada numa *thread* (ou linha de execução) à parte e é responsável por ficar monitorando a condição de transição de estados. Ao verificar-se uma condição verdadeira, uma mudança de estados ocorre, alterando o objeto ativo da classe MonitorInstanciaServico. A cada mudança de estado, uma notificação é enviada a todos os clientes que manifestaram interesse nesse tipo de evento, para essa instância de serviço, pois isso impacta a sincronização e fluxo de controle dos processos virtuais que dependem dessa instância de serviço.

6.1.5. Categoria Gerenciador de Processos Virtuais

A categoria Gerenciador de Processos Virtuais tem como objetivo permitir a gerência da execução de instâncias de processos virtuais (Figura 6.13).

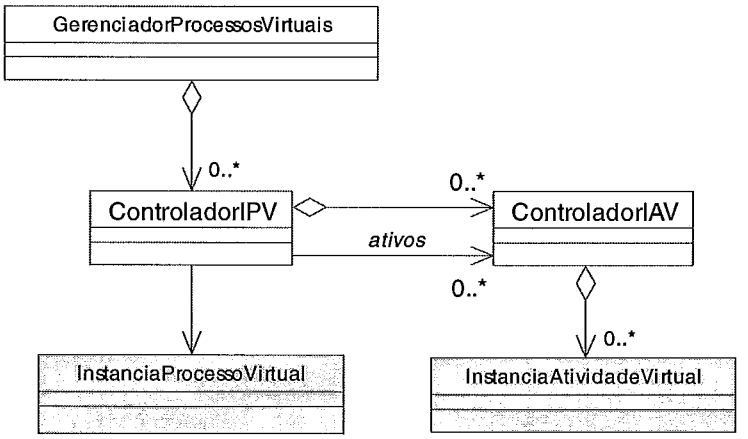

Figura 6.13 – Categoria Gerenciador de Processos Virtuais

A classe Gerenciador de Processos Virtuais (GerenciadorProcessosVirtuais) funciona como o coordenador central dessa categoria. Entre outras funções, ela é encarregada de atender a solicitações de criação de novas instâncias de processos virtuais.

Toda vez que uma instância é criada, ela é encapsulada num objeto da classe Controlador de Instância de Processo Virtual (ControladorIPV). Automaticamente, são criados tantos objetos Controladores de Instâncias de Atividades Virtuais (ControladorIAV) quanto houver atividades definidas para esse processo. Cada objeto ControladorIAV encapsula uma instância de atividade virtual. Ao serem criadas, essas instâncias de atividades ficam num estado chamado "pronto", que significa que a instância já foi criada, mas ainda não foi iniciada sua execução.

O GerenciadorProcessosVirtuais, então, consulta a definição do processo para saber qual é sua primeira atividade e torna ativa a instância de atividade correspondente. Num determinado instante, pode haver várias instâncias de atividades virtuais ativas para uma mesma instância de processo virtual. Isso é expresso na associação ativos entre ControladorIPV e ControladorIAV.

Toda vez que o Gerenciador de Processos Virtuais recebe uma notificação de alteração de estado em uma instância de serviço, verifica se esse fato ocasiona alguma transição em alguma instância de processo. Se isso for verdadeiro, a nova instância de atividade torna-se ativa (se ainda não estava) e seu controlador é acrescentado à lista de ativos do ControladorIPV. A instância de atividade origem da transição pode permanecer ativa também, ou caso tenha atingido o estado "completo" ou "terminado". Caso contrário, seu controlador será retirado da lista de ativos de ControladorIPV correspondente.

Uma atividade pode possuir uma transição para uma atividade fim. Quando a condição de transição para a atividade fim ocorre, ou seja, é alcançada a atividade fim, a execução do processo virtual termina e este entra no estado "completo".

Vale a pena apresentar outro aspecto de projeto relativo a processos virtuais. No Capítulo 5, foi mostrado que uma instância de atividade possui um relacionamento com um objeto da classe Representante de Serviço (RepresentanteServico), como pode ser observado na Figura 6.14.

A classe representante de serviço, como o nome indica, permite que seus objetos ajam como representantes locais (ao processo virtual) de instâncias de serviços que são executadas em outros espaços. Esse critério de projeto reflete outro uso do *design pattern Proxy*, designado *Remote Proxy* por GAMMA et al. (1995), ou ainda, *Ambassador*, por COPLIEN (1992). Embora a estrutura seja a mesma apresentada na Figura 6.8, seu

propósito é outro: como dito anteriormente, agir como um representante local para um objeto em um espaço de endereço diverso.

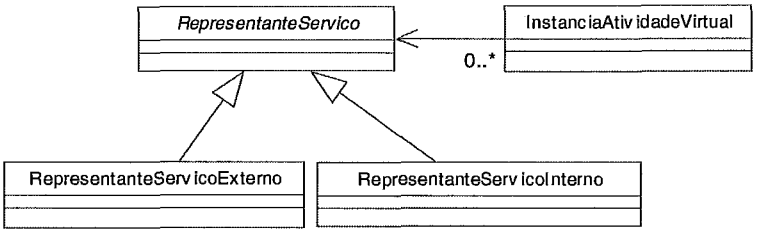

Figura 6.14 – Associação entre Instância de Atividade Virtual e Representante de Serviço

O representante possui os mesmos métodos que a instância de serviço verdadeira e oferece as mesmas funcionalidades. Portanto, na verdade, para a classe Instância de Atividade Virtual isso é transparente, ou seja, tudo se passa como se ela estivesse referenciando o objeto real e não um representante.

6.1.6. Categoria Gerenciador Central

A categoria Gerenciador Central (Figura 6.15) eqüivale ao "sistema nervoso central" do ambiente, funcionando como um coordenador de atividades geral. Entre suas funções estão receber solicitações de clientes, tanto externos quanto internos, acessar outros servidores e gerenciar o acesso a dados feitos pelos demais componentes gerenciadores.

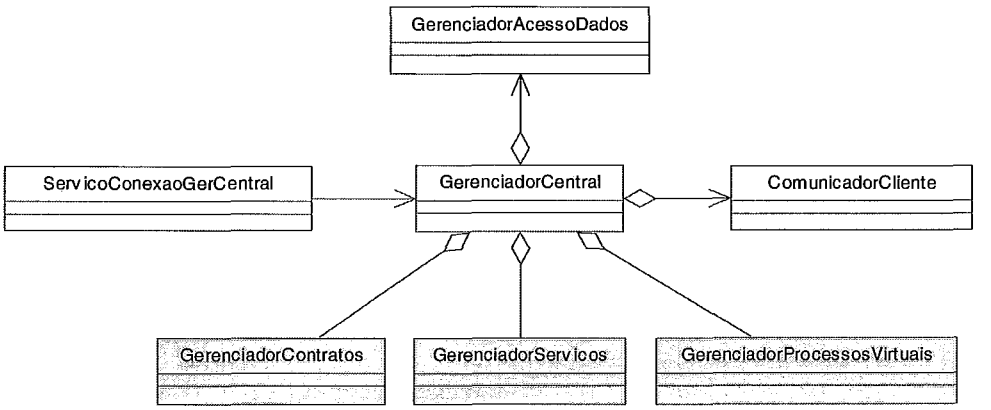

Figura 6.15 – Gerenciador Central

A classe principal dessa categoria é a Gerenciador Central (GerenciadorCentral). Se a categoria como um todo representa o "sistema nervoso central", essa classe é o "cérebro" do ambiente. Ela possui instâncias (uma de cada) de todos os outros gerenciadores e despacha para eles as solicitações que recebe, de acordo com seu tipo. Esses gerenciadores,

então, executam realmente a solicitação. Outra função sua é enviar notificações a outras máquinas sobre eventos de importância ocorridos.

O acesso de clientes ao Gerenciador Central é feito através do Serviço de Conexão ao Gerenciador Central (classe ServicoConexaoGerCentral). Na verdade, essa classe é disponibilizada como um *web service* no servidor de aplicações, ou seja, ela é um serviço acessível via *web*. Os clientes a acessam através do protocolo SOAP (W3C-SOAP, 2001). Como SOAP é um protocolo neutro, os clientes podem ser executados em qualquer plataforma, codificados em qualquer linguagem de programação.

O Gerenciador Central, além de receber solicitações, também precisa se comunicar com outras máquinas, agindo como cliente delas. Isto ocorre em algumas circunstâncias, como: quando um processo virtual é encenado localmente e é preciso iniciar e controlar a execução de serviços externos; quando eventos importantes ocorrem nessa máquina e notificações precisam ser enviadas. A comunicação com outras máquinas (em outras organizações) é feita por meio de uma classe auxiliar, chamada Comunicador Cliente (ComunicadorCliente).

Embora haja uma forma de comunicação padrão (*default*) entre as máquinas das organizações (através de *web services*), cada organização tem liberdade para determinar sua preferência de comunicação. Além disso, também é possível utilizar serviços oferecidos por outros ambientes. Por isso, a escolha do mecanismo de comunicação deve ser feita em tempo de execução, de acordo com a organização a ser contactada.

O *design pattern Strategy* (Figura 6.16) permite que uma determinada estratégia seja determinada em tempo de execução. Isto é, ele é aplicável a situações onde uma estratégia não pode ser definida antecipadamente, como acontece no caso da escolha da forma adequada de comunicação.

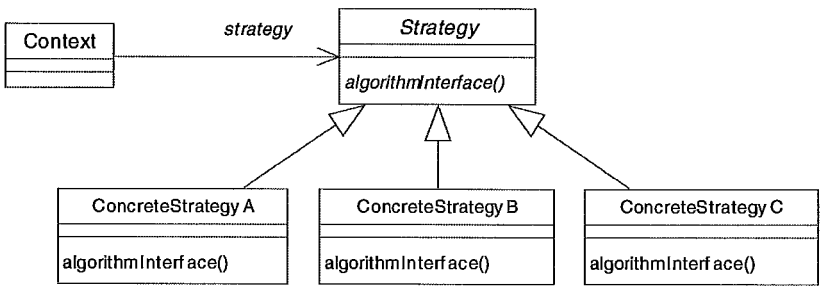

Figura 6.16 – *Design Pattern Strategy* (baseado em GAMMA et al., 1995)

A Figura 6.17 apresenta a classe ComunicadorCliente e algumas de suas especializações, que contemplam as tecnologias de comunicação JAX-RPC, Java RMI e IIOP-CORBA. Novos modos de comunicação podem ser adicionados especializando-se a classe Comunicador Cliente.

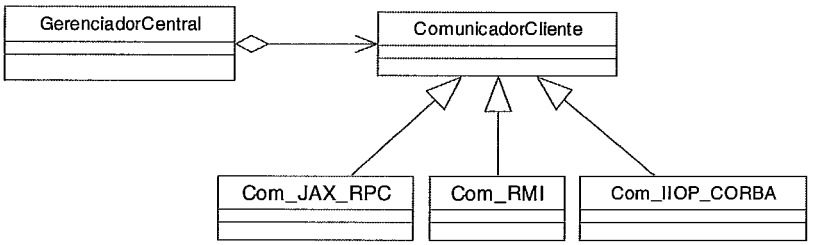

Figura 6.17 – Comunicador Cliente e suas Especializações

A escolha do Comunicador *default* é feita no arquivo de configuração do Gerenciador Central. Foi adotada a linguagem XML (W3C-XML, 2000) para especificação desse arquivo, que se chama config.xml. O Gerenciador Central, ao ser iniciado, lê a *tag* do arquivo config.xml que indica o tipo de comunicador escolhido e instancia o objeto correspondente.

O Gerenciador de Acesso a Dados (GerenciadorAcessoDados) possui o propósito de coordenar o acesso, tanto de leitura quanto de escrita, à base de dados do ambiente. Seu projeto é semelhante ao projeto do Comunicador Cliente. As informações do modelo de dados, apresentadas no Capítulo 5, podem estar armazenadas de diversas formas. Portanto, diversos mecanismos podem ser utilizados para acessar esses dados. Entre as opções estão sistemas gerenciadores de bancos de dados relacionais (SGBDR), sistemas gerenciadores de bancos de dados orientados a objetos (SGBDOO) e objetos serializados Java. Essas opções de mecanismos foram previstas no projeto, como se pode perceber na Figura 6.18.

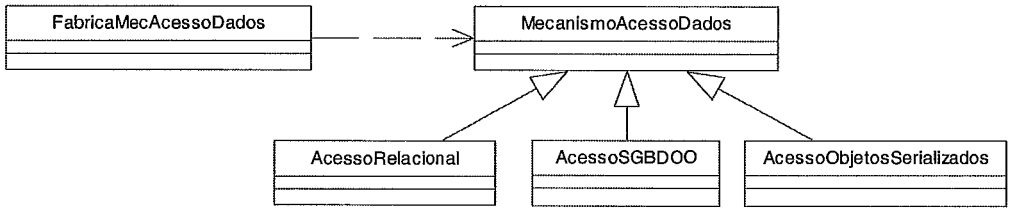

Figura 6.18 – Projeto do Acesso à Base de Dados

A classe Fábrica de Mecanismo de Acesso a Dados (FabricaMecAcessoDados), à semelhança da classe Fábrica de Comunicador Cliente, é responsável por instanciar um dos

mecanismos de acesso a dados disponíveis. Consultando a *tag* correspondente no arquivo config.xml, essa classe descobre qual a instância correta a ser instanciada.

A classe FabricaMecAcessoDados é utilizada pela classe GerenciadorAcessoDados. Após receber a instância de mecanismo de acesso a dados conforme a configuração, uma associação é estabelecida entre GerenciadorAcessoDados e MecanismoAcessoDados, como se pode observar na Figura 6.19.

Figura 6.19 – Projeto do Acesso à Base de Dados

No projeto do acesso aos dados, repetindo novamente o que ocorre no projeto do Comunicador Cliente, novos mecanismos de acesso a dados podem realizados. Basta especializar-se a classe MecanismoAcessoDados e alterar (ou melhor, especializar) a classe FabricaMecAcessoDados.

O Gerenciador Central possui métodos que permitem que clientes externos e internos solicitem a ele serviços, além de métodos para envio de notificações. Esses métodos se baseiam na interface 4 da WfMC (*Interoperability*), com a seguinte diferença: enquanto a interface 4 lida com processos, aqui a unidade de acesso é o serviço. Na primeira versão do ambiente, foi adotado um mecanismo simples de autenticação, baseado em conta (*login*) e senha. Logo, todos os métodos possuem como parâmetros a conta e a senha da organização, que foram omitidos da descrição a ser feita a seguir, para não sobrecarregá-la. Os métodos foram agrupados em métodos de acesso externo, notificações e métodos de acesso interno. Os métodos de acesso interno são utilizados pelas ferramentas acessórias e, obviamente, só são executados se a conta e senha forem da própria organização.

- **Acesso Externo:**

- conecta (): permite ao cliente se conectar a uma máquina, fornecendo sua conta e senha. O Gerenciador Central irá validar o acesso (ou não) e retornar uma mensagem informando o status da conexão.
- desconecta (): permite ao cliente finalizar uma conexão.

- criaInstanciaServico (servico): recebe pedido de uma máquina remota para a instanciação de um serviço, especificado no parâmetro servico, nessa máquina.
- executaInstanciaServico (servico): recebe pedido de um cliente para iniciar a execução de uma instância de serviço (parâmetro servico) nessa máquina.
- obtemObjetoInstanciaServico (instanciaServico, objeto): através desse método clientes podem obter dados extraídos de serviços durante sua execução. Os parâmetros instanciaServico e Objeto indicam em qual objeto, de que instância de serviço, o cliente está interessado.
- obtemEstadoInstanciaServico (instanciaServico): retorna a um cliente o estado atual em que se encontra uma determinada instância de serviço (parâmetro instanciaServico) obtemHistorico (instanciaServico): por meio desse método, o cliente obtém um registro com todo o histórico de eventos registrados para a instância de serviço passada como parâmetro.
- executaOperacao (instanciaServico, operacao): executa uma operação sobre uma instância de serviço que está sendo executada nessa máquina, a pedido de outra máquina.
- listaServicos (): lista todos os serviços disponíveis para esta organização, conforme especificado no contrato.
- listaInstanciasServico (): lista todas as instâncias de serviço, solicitadas por essa organização, que estão em execução no momento.
- cancelaInstanciaServico (instanciaServico): cancela a execução de uma instância de serviço, liberando todos os recursos alocados para ela.
- registraInteresseEventos (instanciaServico, listaEventos): qualquer organização pode se registrar como observadora de determinados eventos gerados pela execução de uma instância de serviço, desde que tal permissão haja sido especificada no contrato do serviço. Através desse método, uma organização indica quais eventos, de qual instância de serviço interessam a ela.
- cancelaInteresseEventos (instanciaServico, listaEventos): cancela o interesse em receber determinados eventos sobre a execução da instância de serviço.

➢ **Notificações**

- objetoAlterado (instanciaServico, objeto): indica às organizações interessadas nessa instância de serviço que dados (objeto) relacionados a uma instância de serviço sofreram alteração.
- estadoInstanciaServicoAlterado (instanciaServico, novoEstado): houve uma mudança de estado em uma instância de serviço e isso é notificado às organizações interessadas.

➢ **Internos**

- criaInstanciaProcessoVirtual (processoVirtual): cria uma instância a partir da definição de um processo virtual (passado como parâmetro) e instancia suas atividades para os agentes concretos.
- iniciaInstanciaProcessoVirtual (instanciaProcessoVirtual): inicia a execução de uma determinada instância de processo virtual.
- suspendeInstanciaProcessoVirtual (instanciaProcessoVirtual): suspende a execução de uma determinada instância de processo virtual.
- retomaInstanciaProcessoVirtual (instanciaProcessoVirtual): retoma a execução de uma determinada instância de processo virtual, após ela ter sido suspensa.
- terminaInstanciaProcessoVirtual (instanciaProcessoVirtual): termina a execução de uma determinada instância de processo virtual.
- monitoraInstanciaServico (ferramenta, instanciaServico): registra a própria ferramenta como receptora de eventos relativos a uma dada instanciaServico.
- enviaControleInstanciaServico (instanciaServico): permite controlar uma instância de serviço, isto é, enviar uma invocação de operação sobre ela.

6.2. Projeto das Ferramentas Acessórias

Esta Seção mostra um *framework* simples para a construção das ferramentas acessórias do ambiente. As ferramentas utilizam o mesmo modelo de dados (definido no Capítulo 5) que o restante do ambiente e possuem dois modos de funcionamento.

No primeiro modo, elas funcionam isoladamente do resto do ambiente, armazenando e acessando informações numa base de dados. Nesse caso, a única interação

com o ambiente seria através da base de dados em comum. No segundo modo, a ferramenta lê e registra informações por intermédio do ambiente. Ou seja, ela envia pedidos de leitura ou escrita de informações ao servidor do ambiente, que os processa adequadamente e executa as ações necessárias sobre a base de dados.

A Figura 6.20 apresenta o *framework* de ferramentas. Nela, podem-se observar suas classes, com alguns atributos e métodos. A classe Usuário (Usuario) representa um usuário da ferramenta e serve, principalmente, para armazenar informações que permitam seu acesso (conta e senha). A classe Categoria representa uma categoria de usuários. A princípio, existem três categorias de usuários: "projetista", "participante" e "monitor". Um usuário pode participar de várias categorias ao mesmo tempo e uma categoria, obviamente, possui vários usuários.

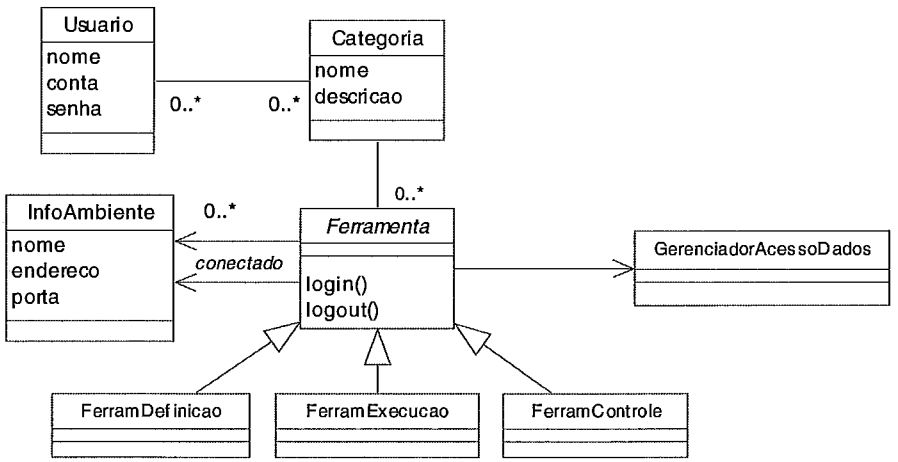

Figura 6.20 – Framework de Ferramentas Acessórias

As ferramentas são desenvolvidas para atender a essas categorias de usuários. Portanto, cada categoria pode possuir diversas ferramentas. Para simplificar o modelo, assumiu-se que cada ferramenta atende a somente uma categoria.

Uma instância da classe Ferramenta representa uma ferramenta acessória propriamente dita. A classe Ferramenta é abstrata, mas três tipos de ferramentas (especializações de Ferramenta) foram definidos *a priori*: ferramenta de definição (FerramDefinicao), ferramenta de execução (FerramExecucao) e ferrametna de monitoramento e controle (FerramControle). Essas ferramentas visam atender às categorias "projetista", "participante" e "monitor", respectivamente. Novas ferramentas podem ser criadas estendendo-se a classe Ferramenta, ou ainda, uma das três ferramentas específicas.

No intuito de atender aos dois modos de funcionamento das ferramentas, duas classes são usadas no *framework*. A primeira, InfoAmbiente, armazena informações necessárias para a conexão da ferramenta a um servidor de ambiente e atende ao primeiro modo de funcionamento. A segunda, GerenciadorAcessoDados, permite o acesso da ferramenta a uma base de dados, semelhantemente ao componente homônimo definido na Seção 6.1.6, atendendo dessa forma ao segundo modo de funcionamento das ferramentas.

6.3. Desenvolvendo Aplicações no Ambiente

Esta Seção tem como objetivo apresentar um roteiro para o desenvolvimento de aplicações utilizando o ambiente. A Seção 6.1 apresenta um conjunto de atividades administrativas que devem ser feitas para configurar o ambiente. Essa etapa pode servir para mais de uma aplicação. Depois, na Seção 6.2, são mostradas as atividades necessárias para o estabelecimento de uma organização virtual e suas aplicações. As atividades foram nomeadas segundo o seguinte critério: as atividades de configuração possuem a estrutura C.n, onde n determina o número da atividade. As atividades de desenvolvimento seguem padrão similar, sendo nomeadas D.n, segundo a mesma lógica.

6.3.1. Configurando o Ambiente

Essa etapa é constituída por atividades opcionais. Elas só serão realizadas caso se queira mudar ou adicionar novos comportamento à infra-estrutura, antes de desenvolver novas aplicações nela. Essas atividades, após terem sido efetuadas, servirão para todas as aplicações.

- **Atividade C.1: Criar novo mecanismo de comunicação**

Novos mecanismos de comunicação podem ser adicionados à infra-estrutura, de modo a permitir a conexão com outros ambientes que disponibilizem seu acesso através de um meio que não tenha sido previamente disponibilizado. Essa atividade, então, tem como objetivo a criação desse novo meio. Isso é feito estendendo-se a classe ComunicadorCliente, conforme sugere a Figura 6.21. Nela, pode-se observar a criação de

um novo mecanismo de comunicação para acesso a objetos no padrão Enterprise Java Beans (EJB).

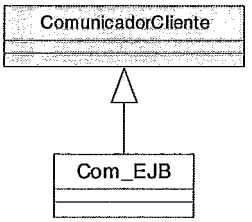

Figura 6.21 – Estendendo o Comunicador Cliente

➢ Atividade C.2: Criar mecanismo de acesso a dados

A base de dados da infra-estrutura pode estar sob diversas formas, como SGBDR, SGBDOO, arquivos serializados Java, etc. Se o administrador do ambiente quiser utilizar alguma outra forma, isso pode ser feito através da criação de um novo mecanismo de acesso a dados, estendendo-se a classe MecanismoAcessoDados. A Figura 6.22 mostra, como exemplo, a criação de um novo tipo de acesso, AcessoArquivoASCII, que permite o acesso a informações armazenadas em arquivos simples ASCII, de forma estruturada.

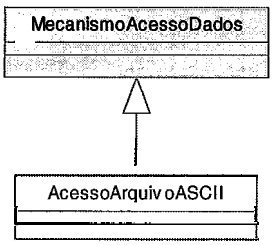

Figura 6.22 – Estendendo o Mecanismo de Acesso a Dados

➢ Atividade C.3: Determinar itens *default* da infra-estrutura

Nessa atividade o administrador do ambiente especifica dois itens fundamentais para o seu funcionamento: o mecanismo de acesso à base de dados (relacional, OO, arquivo serializado, etc.) e a comunicação padrão. As informações de configuração ficam armazenadas no arquivo config.xml. O Gerenciador Central, ao ser iniciado, lê esse arquivo e instancia o restante da infra-estrutura de acordo com os parâmetros especificados nesse arquivo.

➢ **Atividade C.4: Estender gramática de expressões**

Caso se deseje adicionar novos *tokens* à gramática que define a linguagem usada para se especificar as expressões que condicionam as transições entre estados, é preciso implementar uma de duas interfaces: a interface BooleanExp ou IntExp, para expressões booleanas ou aritméticas, respectivamente. Pode-se ainda, definir outros tipos de expressões, desde que respeitem a mesma interface (que define o método interpret), mas isso é menos provável. A gramática estendida poderá ser utilizada em diversas aplicações.

6.3.2. Desenvolvendo uma Aplicação

Para desenvolver uma aplicação no ambiente, podem ser utilizadas duas abordagens distintas. Na primeira, *bottom-up*, primeiramente definem-se os tipos e serviços e depois as organizações e os processos virtuais que utilizam os serviços definidos. Na segunda, *top-down*, primeiramente define-se a estrutura da organização virtual e seus processos e só depois se começa a definição e implementação dos serviços. As atividades que compõe as duas abordagens são as mesmas.

➢ **Atividade D.1: Criar tipos de esquemas de estados**

A interface de um serviço é determinada pelo tipo de esquema de estados que ele possui. Assim, para se definir um serviço como especialização de outro, é preciso criar um novo tipo de esquema de estados especializando o tipo que o serviço pai possui. Para criar um novo serviço independente de qualquer outro criado anteriormente, basta criar um tipo de esquema de estados que especializa o tipo básico, WfMC.

Depois de criado o tipo de esquema de estados, seus novos estados devem ser definidos. Cada novo estado especializa um estado presente no tipo esquema de estados da qual o novo tipo herda as características. O mais comum é se especializar o estado "executando", embora seja permitido especializar quaisquer outros, como "suspenso" e "completo", por exemplo.

A seguir, devem ser definidas novas transições de estados, entre os novos estados, assim como entre eles e os estados antigos.

O passo seguinte consiste em definir as assinaturas das operações que poderão ser invocadas sobre serviços que implementem este tipo. Essas operações servirão para empresas que contratem o serviço possam controlá-lo. Ao definir-se uma assinatura de operação, indica-se seu nome, o tipo de objeto que a operação retorna e qual transição de estados ela causa. É preciso definir também o tipo de objeto de cada parâmetro necessário para sua invocação. Por fim, indica-se a localização física da classe que implementa a operação.

Feito isso, passa-se à implementação das classes para cada assinatura de operação definida. A classe da nova operação deve especializar uma das classes definidas previamente: "OpExecuta", "OpSuspende", "OpRetoma" ou "OpTermina". A implementação da classe contém as ações necessárias de serem realizadas para a execução da operação, ou seja, codifica-se como a invocação da operação afeta o processo interno. Contudo, essa codificação utiliza as primitivas do *Workflow* Abstrato e independe de detalhes de produtos específicos. A Figura 6.23 mostra um exemplo de criação de nova operação, especializando-se a classe OpExecuta.

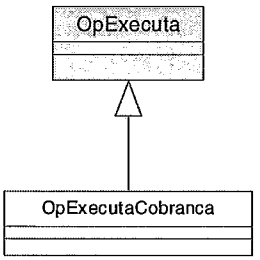

Figura 6.23 – Estendendo uma Operação

➢ Atividade D.2: Definir serviço

A definição de serviços é feita nessa atividade. Primeiro, são descritos os atributos gerais do serviço, como nome, descrição, parâmetros de qualidade de serviço (QoS) e objetos (estruturas de dados) de entrada e saída. Escolhe-se também o tipo de esquema de estados para esse serviço e o agente que ele mapeará (aplicação ou processo). Por fim, descreve-se a sua interface, que será usada para sua contratação.

➢ Atividade D.3: Criar organização virtual

Essa atividade consiste em criar a organização virtual, a partir das outras

organizações reais e/ou organizações virtuais já existentes. Escolhe-se que organizações comporão a nova organização virtual. Caso seja uma organização virtual do modelo centralizado, é determinada sua organização líder.

Após ser definida a composição da organização virtual, deve-se criar o contrato que a regulamentará. Especificam-se os critérios gerais do contrato, em linguagem natural, e as regras de utilização dos serviços nessa organização, como quais empresas têm direito de utilizá-lo e quantas vezes, por exemplo.

- **Atividade D.4: Definir processo virtual**

Nessa atividade é definido um novo processo virtual. Primeiro, são fornecidas informações básicas sobre o processo, como nome, descrição, etc. Depois, definem-se novas atividades virtuais. Em cada atividade virtual, deve-se determinar o tipo do serviço que a implementará (tipo de esquema de estados). Caso se queira filtrar os serviços que poderão ser usados para a realização da atividade, isso deve ser especificado no filtro da atividade. Por exemplo, pode-se definir que uma dada atividade só pode ser realizada por serviços externos, nunca internos. Eventualmente, pode-se estabelecer um serviço *default* para a implementação da atividade. Se isso não for feito nesse momento, poderá ser realizado no instante da instanciação do processo. Para ligar uma atividade a outra(s), são definidas transições. Em cada transição determina-se a condição que a desencadeia.

6.3.3. Processo de Criação de Aplicações

Será apresentada, agora, uma definição gráfica de processo que ilustra como as atividades definidas nas Seções 6.3.1 e 6.3.2 podem ser encadeadas, segundo a abordagem *bottom-up* (Figura 6.24). O processo tem início executando-se as atividades C.1 e C.2. Ambas precisam ser terminadas antes do início de C.3. C.4. é realizada na seqüência. Terminada a configuração, pode-se iniciar o desenvolvimento da aplicação (atividades D.1 a D.4). Essas atividades são realizadas seqüencialmente. No entanto, algumas delas podem ser realizadas mais de uma vez antes de se passar para a próxima. São elas: D.1, D.2 e D.4. O processo torna-se completo após a realização da última iteração da atividade D.4.

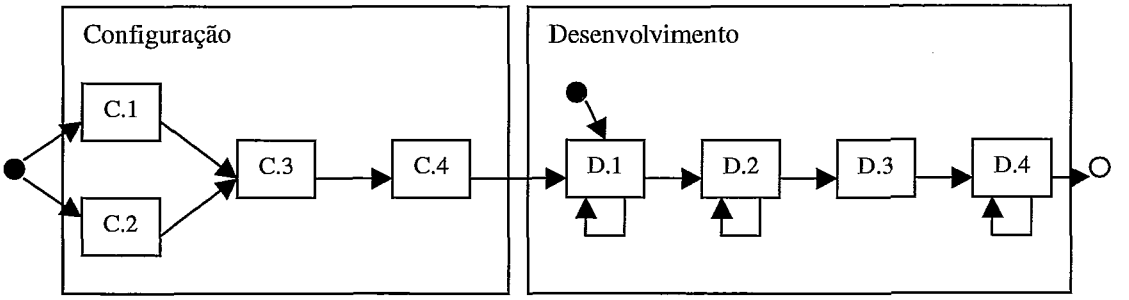

Figura 6.24 – Processo de Criação de Aplicações

Se o ambiente já estiver preparado e configurado adequadamente para a nova aplicação, pode-se começar o processo pelo desenvolvimento, isto é, não é preciso efetuar as atividades relativas à sua configuração.

6.4. Conclusões Preliminares

Este capítulo apresentou o projeto da infra-estrutura do ambiente. Foram discutidas as principais decisões de projeto tomadas durante sua elaboração, juntamente com as soluções adotadas. Diversos *design patterns* foram utilizados para tornar o projeto mais flexível e extensível. Em seguida, foi apresentado o *framework* para desenvolvimento das ferramentas acessórias ao ambiente. Por último, um roteiro de criação de aplicações foi mostrado, destacando as atividades de configuração, desenvolvimento e como elas são coordenadas, através de um processo.

Algumas características da infra-estrutura se destacam, inclusive entre os demais sistemas propostos na literatura: a possibilidade de definição de expressões complexas para mapear processo a serviços, o apoio à utilização de diversos WfMSs comerciais, a possibilidade de escolha da base de dados, a escolha do mecanismo de comunicação, o uso de padrões abertos para web (*web services*) e o gerenciamento dinâmico de serviços e processos virtuais.

Depois de ter sido apresentado o projeto da infra-estrutura, será apresentada sua implementação no capítulo seguinte (Capítulo 7). Também será visto um exemplo de aplicação criada no ambiente, onde ficarão mais claros os passos que constituem o processo de criação.

7 – Implementação e Exemplo de Aplicação

Após ter sido apresentado o projeto da infra-estrutura no Capítulo 6, esse capítulo apresenta algumas questões de implementação dessa infra-estrutura, mostrando algumas das tecnologias empregadas, assim como produtos comerciais utilizados.

Será mostrado nesse capítulo também um exemplo de aplicação desenvolvido no ambiente, com o intuito de demonstrar sua funcionalidade e ilustrar como se pode realizar o processo de criação de aplicações.

A implementação será apresentada na Seção 7.1. A Seção 7.2 mostra o exemplo de aplicação.

7.1. Visão Geral da Implementação

A codificação da infra-estrutura foi feita utilizando-se a linguagem de programação Java (Java 2 SDK, Standard Edition, versão 1.3.x). A escolha dessa linguagem se deve a algumas das características que a tornaram tão bem sucedida e amplamente utilizada. Em primeiro lugar, por apresentar sólidos conceitos de orientação a objetos, o que facilita a implementação de um projeto elaborado com essa tecnologia; por sua portabilidade: escreve-se o código uma vez e pode-se executá-lo em diversas plataformas, podendo atender assim a diversas empresas, que é um dos objetivos do trabalho; por possuir vasta biblioteca de classes disponível, o que facilitou o desenvolvimento. Há classes, por exemplo, para acesso a bases de dados relacionais, processamento XML e para desenvolvimento de *web services*.

Como sistema de gerenciamento de *workflow* (WfMS), foi utilizado o software MQSeries Workflow, da IBM (IBM-MQWF, 2001c), versão 3.3.x. Este software será descrito em mais detalhes na Seção 7.1.1.

O armazenamento e gerenciamento de dados foram feitos utilizando-se o SGBD relacional DB2, da IBM (IBM-DB2, 2002), versão 7.2 Personal Edition. O modelo de dados orientado a objetos apresentado no Capítulo 5 foi adaptado para originar o esquema de dados (relacional) inserido no DB2. Esse processo foi feito seguindo recomendações

encontradas em livros de métodos orientados a objetos, como os de JACOBSON et al. (1992) e RUMBAUGH et al. (1994).

O acesso ao banco, no entanto, foi implementado utilizando-se a tecnologia JDBC (Java Database Conectivity), versão 2.0. Dessa forma, evita-se a restrição a somente um produto específico de banco de dados e pode-se, a qualquer momento, trocar o SGBD sem que isso afete o restante da infra-estrutura. Basta que o novo SGBD possua um *driver* JDBC ou ODBC.

A fim de se utilizar o conceito de *web services*, a arquitetura teve que contar com um servidor de aplicações, responsável por oferecer, via *web*, páginas html e aplicações em Java, como *servlets* e páginas JSP. O servidor de aplicações adotado foi o Jakarta Tomcat, versão 4.0.x (TOMCAT, 2002) que é um software gratuito e de código aberto, desenvolvido pelo projeto Jakarta (JAKARTA, 2002), da Fundação Apache (*Apache Software Foundation*) (APACHE, 2002). Essa fundação só produz software com essas características. Este servidor de aplicações é robusto e atende bem a necessidades básicas de software desenvolvido para *web*, sendo amplamente utilizado em ambiente acadêmico e mesmo no mercado (em empresas de porte pequeno/médio), para fins de desenvolvimento e, às vezes, em produção.

A implementação do protocolo de comunicação SOAP (W3C-SOAP, 2002) foi feita através de uma ferramenta fornecida pela Sun para a plataforma Java, chamada JAX-RPC (Java API for XML-based RPC) (JAX-RPC, 2002). Como o nome sugere, é uma API para construção de *web services* e clientes utilizando chamadas remotas de procedimentos (RPC) e XML. Um mecanismo RPC permite que clientes executem procedimentos em outros sistemas, num modelo distribuído cliente/servidor. No JAX-RPC, a chamada remota é codificada em SOAP. Dessa forma, o implementador só precisa se preocupar com a programação Java. A parte específica do protocolo SOAP fica "escondida" por trás da API. Uma vantagem do uso do JAX-RPC é que clientes desenvolvidos com esse mecanismo podem acessar *web services* desenvolvidos em outras linguagens e vice-versa, por se basear em padrões abertos (SOAP, HTTP e WSDL).

Por fim, como utilitário para compilação e execução das classes, em particular no uso do JAX-RPC, foi utilizada a ferramenta Ant (ANT, 2002), também desenvolvida pelo projeto Jakarta. Essa ferramenta, destinada à programação Java, assemelha-se à ferramenta

"make", muito utilizada em ambiente Unix para programação em C/C++. Através da criação de um arquivo de configuração (em XML) contendo as instruções de compilação, uma série de ações acessórias (por exemplo, mover código fonte do diretório origem para o diretório de compilação, indicar diretórios onde estão as bibliotecas de classes necessárias à compilação, mover arquivo final do *web service* para o diretório do servidor de aplicações, etc.) podem ser realizadas, facilitando assim o trabalho de desenvolvimento.

7.1.1. MQSeries Workflow

O MQSeries Workflow (MQWF) é um software de gerenciamento de workflow desenvolvido pela IBM. Para seu funcionamento, o MQWF depende de dois outros produtos: O MQSeries, software de gerenciamento de filas, utilizado para realizar a comunicação entre componentes clientes e servidores e entre servidores; e o DB2, SGBD utilizado para armazenar os esquemas de modelagem (buildtime) e de encenação de processos (runtime).

Programas podem acessar o sistema via API. Existem APIs para as linguagens Java, Visual Basic (ActiveX), C, assim como para a ferramenta Lotus Notes. A API Java é bastante completa e permite realizar tanto funções de definição, como criar uma nova definição de processo, por exemplo, quanto de execução e monitoramento. Esse foi um dos principais motivos para a escolha do MQSeries Workflow para realizar esse trabalho, uma vez que a linguagem Java foi adotada para sua programação.

Uma breve descrição da arquitetura do MQWF será feita a seguir. Logo após, serão mostradas as ferramentas de modelagem de processos e de tratamento de listas de trabalho que fazem parte do MQSeries Workflow.

Arquitetura

Os componentes do MQSeries Workflow foram projetados para uma arquitetura multi-camadas (IBM-MQWF, 2001c). A Figura 7.1 mostra os principais componentes numa arquitetura em três camadas.

Na camada um, encontra-se o cliente MQWF, que pode ser um programa acessando o servidor ou ainda alguma ferramenta responsável por permitir aos usuários o acesso às

suas listas de trabalhos. A comunicação com o servidor é feita através do MQSeries, utilizando a camada de mensagens do MQWF.

A camada dois contém os componentes servidores e de definição (buildtime) do MQWF. Os componentes servidores são responsáveis por gerenciar a execução de processos e podem estar distribuídos em várias máquinas. Nesse caso, a comunicação entre os diversos servidores é feita utilizando-se o MQSeries.

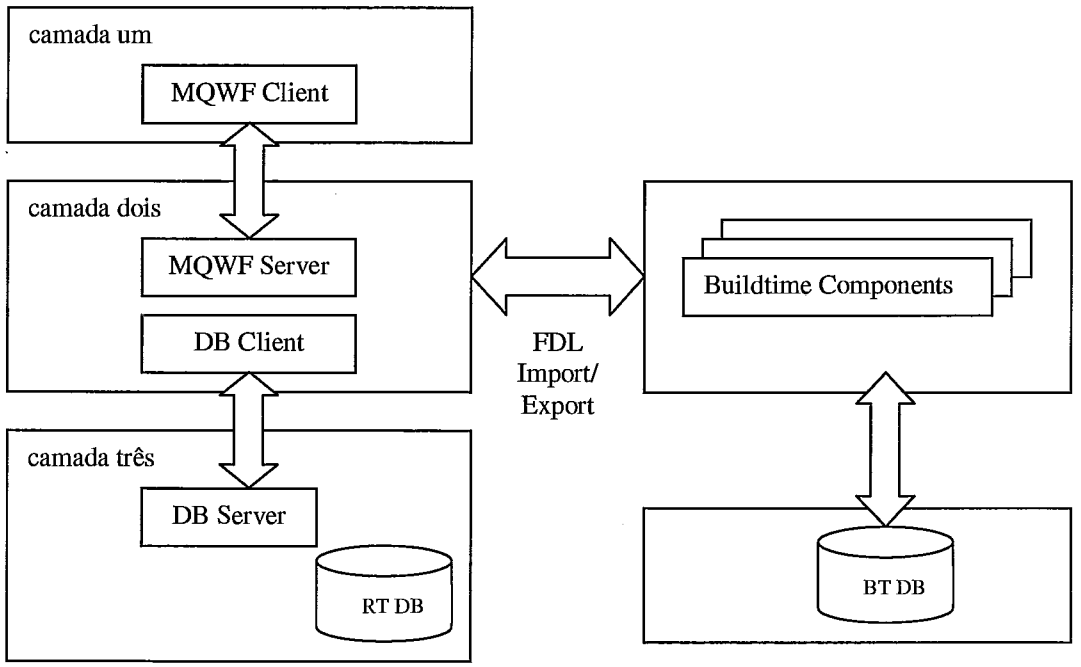

Figura 7.1 – Arquitetura do MQSeries Workflow

A camada três é a camada de banco de dados. O banco de dados registra informações sobre a configuração e execução do MQWF. A comunicação entre cliente e servidor de banco de dados é feita utilizando-se os protocolos oferecidos pelo DB2.

Uma segunda possibilidade de distribuição dos componentes consiste em pôr o banco de dados na mesma máquina onde estão os servidores MQWF (arquitetura em duas camadas). Outra possibilidade existente é utilizar um navegador *web* como cliente. Nessa alternativa, é adicionada uma outra camada (*web*) e a arquitetura fica distribuída em quatro camadas.

Ferramenta de Modelagem (MQSeries Workflow Buildtime)

A ferramenta de modelagem do MQSeries Workflow (Buildtime) permite que sejam criadas as definições de processos que, posteriormente, serão encenados pelo sistema (IBM-MQWF, 2001b).

Entre os itens que podem ser criados estão programas e estruturas de dados, pessoas e papéis e o processo em si. A janela principal da ferramenta (Figura 7.2) é dividida em cinco áreas: uma barra de menu no topo; uma barra de ferramentas logo abaixo desta; uma visão em forma de árvore dos itens criados no canto esquerdo; uma palheta de ferramentas de desenho à direita desta última; e uma área de trabalho.

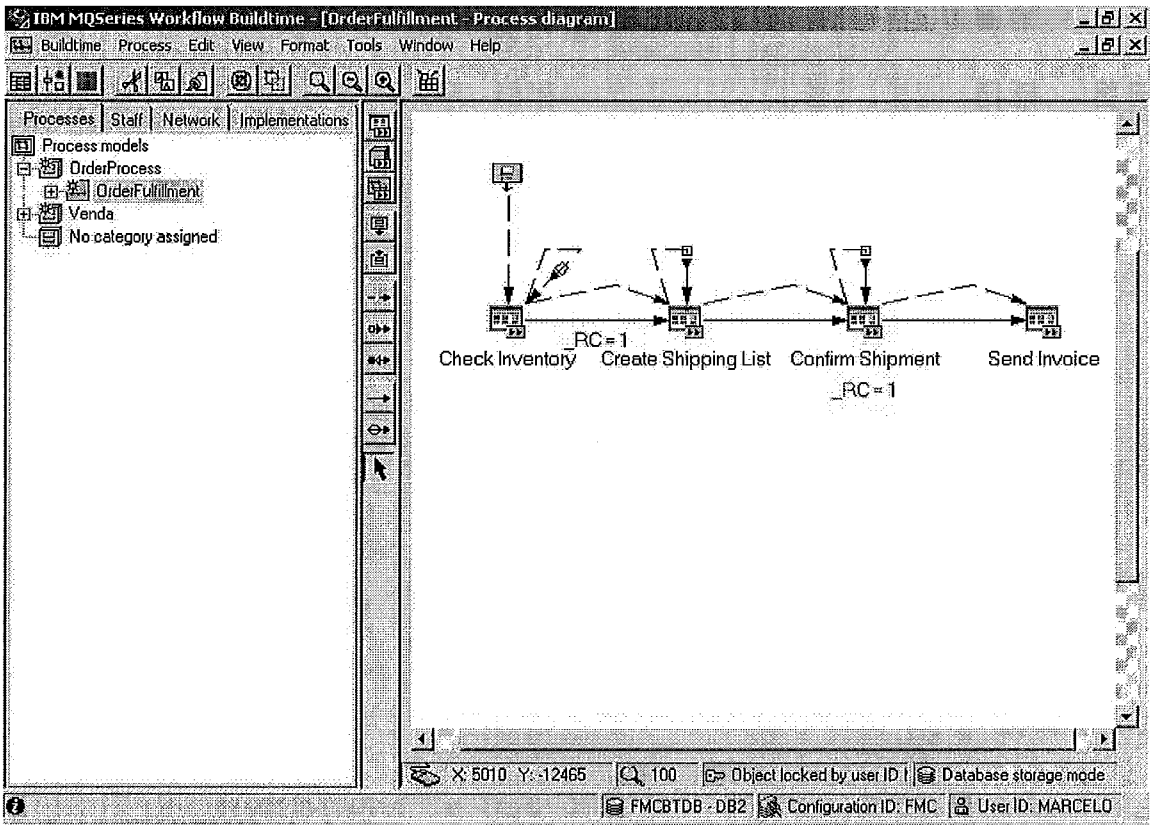

Figura 7.2 – Ferramenta de Modelagem do MQSeries Workflow

A modelagem gráfica é feita escolhendo-se a ferramenta adequada na palheta de desenho (há atividades de programas, atividades de processo, conectores de dados e controle, etc.) e arrastando-a para a área de trabalho. Na Figura 7.2, pode-se observar um processo definido, contendo 4 atividades de programas e conectores de dados e controle entre elas.

Uma possível abordagem para a definição de processos seria começar pela criação das estruturas de dados e programas que seriam utilizadas pelo novo processo. A seguir, definir-se-iam as pessoas e papéis que seriam utilizadas para a execução das atividades. Por último, desenhar-se-ia o processo, pondo as atividades na área de trabalho e ligando-as através de conectores de controle e dados. Em cada atividade, deve ser definido o programa responsável por sua execução, caso seja uma atividade de programa, que é a mais utilizada. Caso seja uma atividade de processo, deve-se definir qual é o processo que a executa.

Ferramenta Cliente (MQSeries Workflow Cliente)

A ferramenta cliente do MQSeries Workflow (IBM-MQWF, 2001a), mostrada na Figura 7.3, permite que usuários possam criar instâncias de processo, iniciar atividades numa lista de trabalho, definir como filtrar e ordenar itens numa lista de trabalho, criar e remover listas de trabalho, modificar o estado de uma atividade e monitorar o progresso de atividades numa instância de processo.

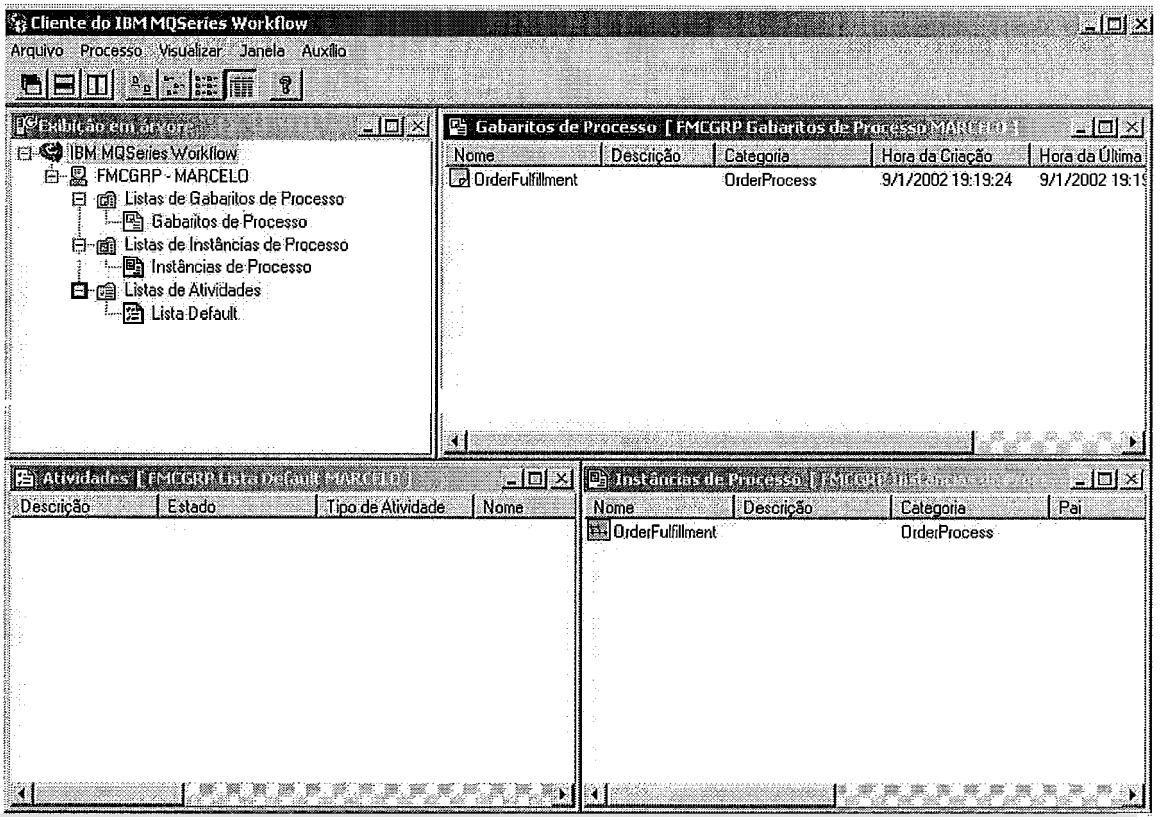

Figura 7.3 – Ferramenta Cliente do MQSeries Workflow

A Figura 7.3 mostra a janela principal da ferramenta cliente. Nela, podem-se observar quatro janelas secundárias. No canto superior esquerdo, pode-se visualizar uma árvore de objetos para o usuário Marcelo, que está conectado no momento. A árvore inclui uma lista de gabaritos (definições) de processos, uma lista de instâncias de processos e uma lista de atividades. No canto superior direito, uma definição de processo ("OrderFulfillment") é mostrada na janela de gabaritos de processos. No canto inferior esquerdo, mostra-se a lista de atividades para o usuário Marcelo. Nota-se que no momento não há nenhuma atividade assinalada a esse usuário. Por fim, na janela inferior direita, há uma lista de instâncias de processos iniciadas por esse usuário. Pode-se ver, então, uma instância do processo "OrderFulfillment" criada e em execução.

7.2. Exemplo de Aplicação

Esta Seção apresenta um exemplo de criação de aplicação de integração de processos. Esse exemplo foi inspirado e adaptado de um caso real, que é o modelo de negócios da empresa fabricante de roteadores e servidores para redes CISCO, situada nos Estados Unidos. O estudo de caso que descreve esse modelo de negócios pode ser encontrado num relatório sobre comércio eletrônico publicado pelo Departamento de Comércio dos EUA (MARGHERIO et al., 1995).

O exemplo consiste no estabelecimento de uma organização virtual, formada por três tipos de organizações, com o objetivo de vender servidores e outros equipamentos de rede. Esses produtos são bastante específicos e vendidos somente sob encomenda. Em linhas gerais, o processo de venda consiste em registrar a venda, encomendar do fabricante o(s) produto(s), configurá-lo(s) e entregá-lo(s) ao cliente. O processo principal (venda) pertence a uma das empresas, que delega algumas de suas atividades às outras duas empresas. A empresa que faz a venda, nesse contexto, é considerada a organização líder da organização virtual. Por isso, o processo virtual é definido e encenado em seu domínio. Admite-se, como exemplo, que as três empresas utilizam o MQSeries Workflow como WfMS onde estão definidos e são encenados seus processos internos.

Antes de apresentar o processo virtual, apresentaremos como a organização que realiza a venda definia seu processo, no modo tradicional, isto é, sem integração automatizada com as demais empresas.

7.2.1. Processo de Venda de Roteadores

A Figura 7.4 apresenta a definição do processo de venda de roteadores da OrganizaçãoA. Embora esteja definido num sistema de *workflow*, esse processo não possui nenhuma integração automatizada com os processos das outras empresas. Nas atividades onde é preciso interagir com as outras organizações, trabalhadores acionados pelo WfMS utilizam meios tradicionais de comunicação, como telefone e fax para realizar essa interação.

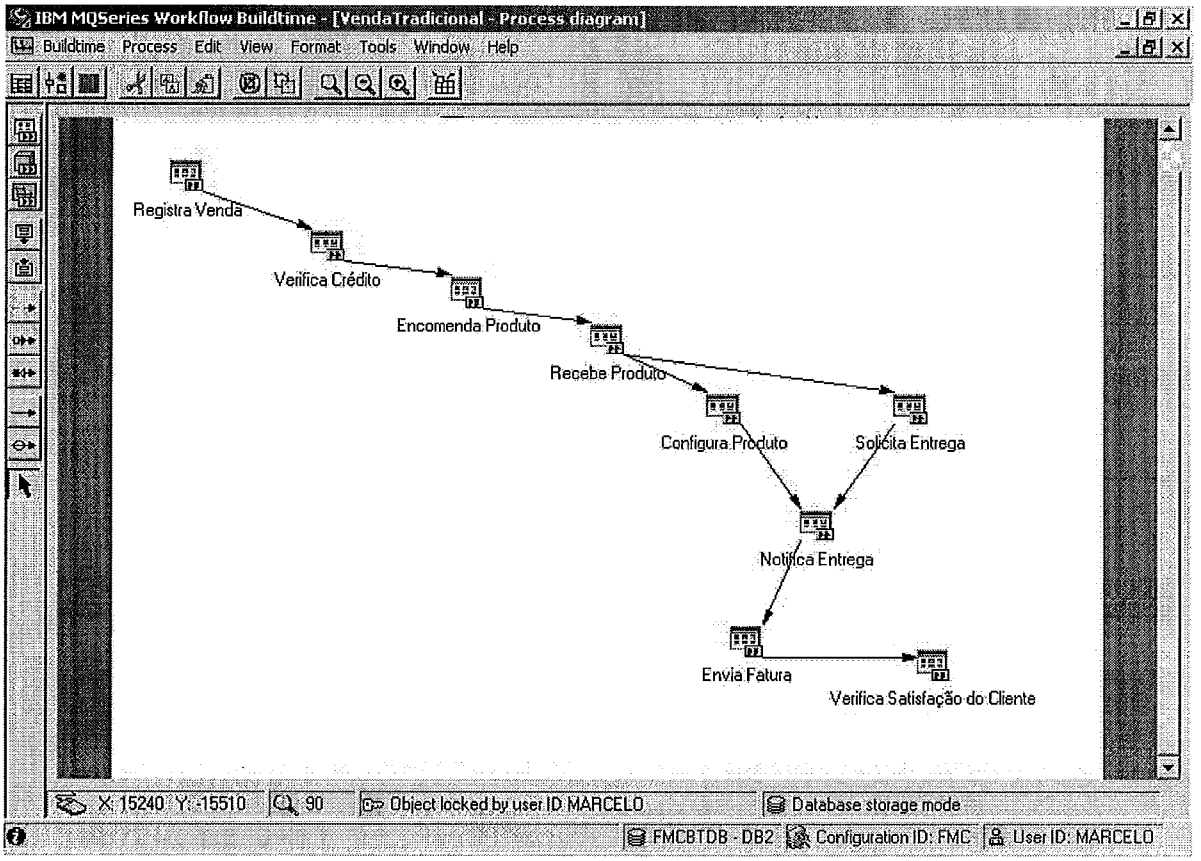

Figura 7.4 – Processo Tradicional de Venda de Produto

Como se pode ver na Figura 7.4, o processo é composto de nove atividades:

1) Registra venda: coleta informações sobre o produto, cadastra cliente e venda, etc.
2) Verifica crédito do cliente: verifica se o cliente possui crédito para comprar tal produto.

3) Encomenda produto ao fabricante: uma pessoa recebe o pedido de venda e envia, por fax, a solicitação do produto.
4) Recebe produto: recebe o produto, proveniente do fabricante, e verifica se está o.k. quanto a critérios de qualidade.
5) Configura: configura o produto para atender às necessidades do cliente. Essa é uma atividade fundamental, pois é onde essa organização pode adicionar valor ao processo, diferenciando-se de suas concorrentes.
6) Solicita entrega à transportadora: solicita, por telefone, a entrega do produto à transportadora. A transportadora avisa quando será feita a coleta do produto e quando este será entregue ao cliente.
7) Envia notificação da data de entrega: um trabalhador verifica a data de entrega assinalada e envia, por e-mail ou correspondência, uma notificação ao cliente contendo a data de entrega.
8) Envia fatura ao cliente: por correspondência, é enviada ao cliente uma fatura.
9) Verifica satisfação do cliente: um funcionário telefona para o cliente alguns dias após a entrega, a fim de verificar sua satisfação quanto ao serviço prestado.

Esse processo, no entanto, poderia ser melhorado, principalmente em relação a aspectos que dizem respeito à interação com as outras empresas. A partir das perspectivas abertas pela utilização do ambiente, um novo processo pôde ser definido, para aproveitar melhor as possibilidades de um processo realmente integrado.

O novo processo (Figura 7.5) possui as mesmas atividades que o tradicional, porém a ordem em que suas atividades são executadas for reorganizada, de forma a aproveitar melhor as novas possibilidades de interação.

No novo processo, logo após o produto ter sido encomendado ao fabricante, pode-se enviar a fatura e notificação de data de entrega ao cliente, baseando-se no prazo de entrega do produto fornecido pelo fabricante. O prazo total máximo para a entrega é conhecido porque o tempo de configuração do produto, obviamente, é conhecido e o tempo máximo de entrega pela transportadora também, pois há um acordo com as transportadoras que as obrigam a entregar o produto em até n dias a partir da solicitação de entrega feita a elas.

Há duas vantagens nessa nova ordenação. A primeira é a satisfação do cliente. Ao receber mais cedo a notificação da data da entrega do produto comprado, há a impressão de que o processo é mais rápido, além de deixá-lo melhor informado em menos tempo. A segunda vantagem, dessa vez para a organização, é poder cobrar o cliente antecipadamente.

É importante ressaltar, no entanto, que a mudança no processo foi feita exclusivamente visando uma melhoria e não por exigência da integração de processos no ambiente. Se a OrganizaçãoA desejasse, poderia manter intacto seu processo e ainda assim a integração seria feita. Obviamente, teria menos benefícios, mas ainda assim seria vantajoso: por exemplo, poder-se-ia reduzir ou otimizar custo de pessoal, eliminando as funções de comunicação com as empresas parceiras feitas por indivíduos.

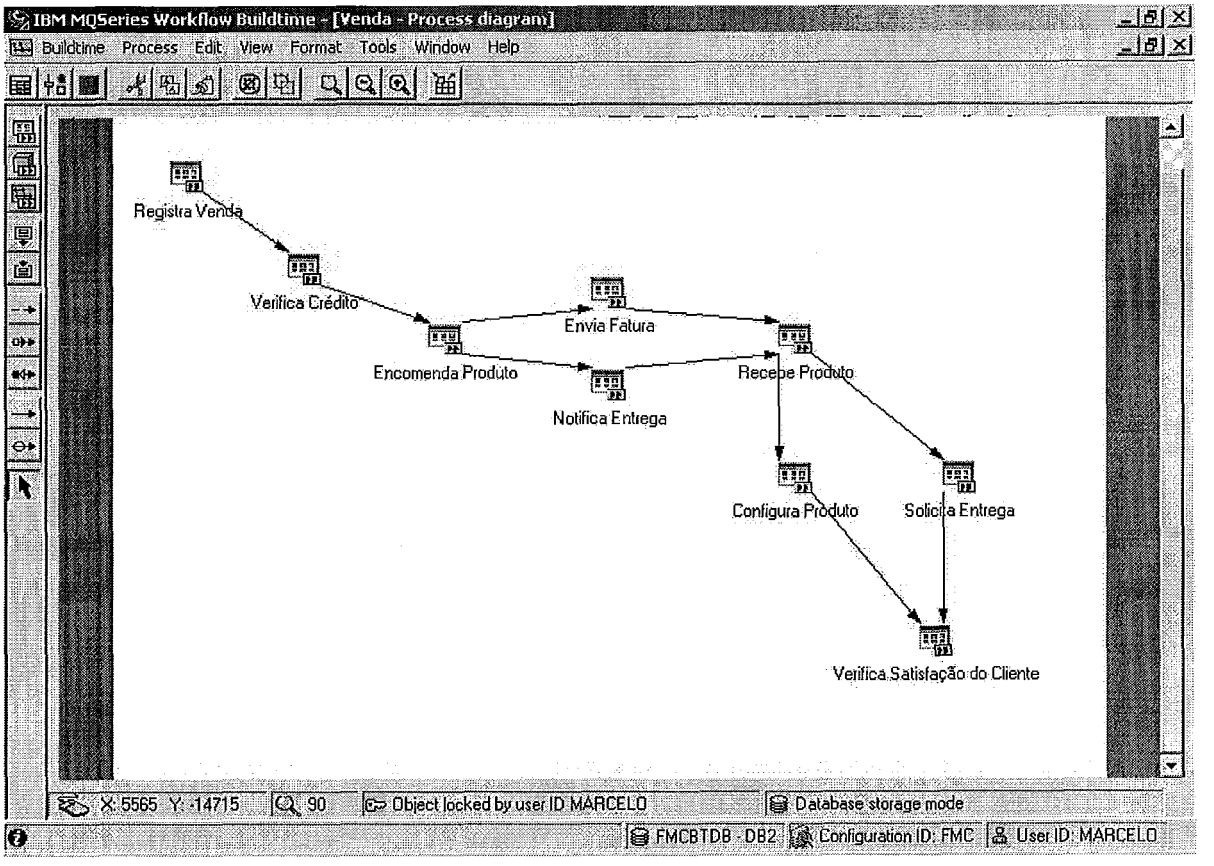

Figura 7.5 – Processo Otimizado de Venda de Produto

7.2.2. Processo Virtual de Venda de Roteadores e Serviços

O processo virtual, que engloba os serviços de Venda (principal), Fabricação e Entrega, pode ser visto na Figura 7.6. Há três atividades virtuais nesse processo, uma para cada serviço. Vale a pena lembrar que várias empresas podem oferecer o mesmo serviço e a escolha da empresa que realmente o executará pode ser feita em tempo de execução, dinamicamente.

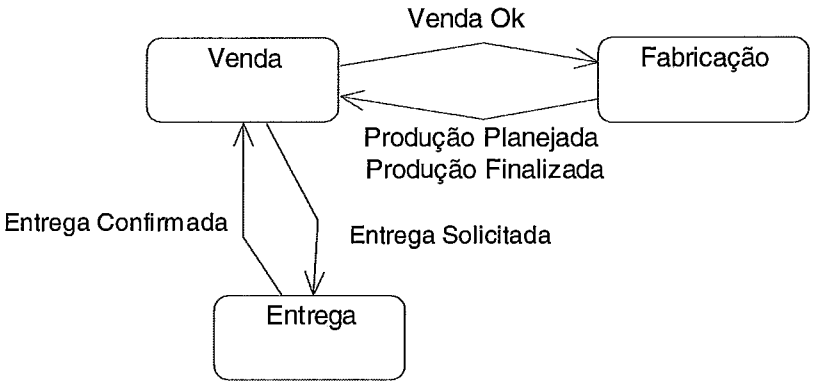

Figura 7.6 – Processo Virtual de Venda

Para a efetivação desse processo virtual, três serviços precisam ser criados: serviço de Venda, serviço de Fabricação e serviço de Entrega. Cada atividade será realizada por um desses serviços. Cada serviço, por sua vez, exige a criação de um novo tipo de esquema de estados, a fim de permitir o controle e monitoramento sobre ele.

O serviço de Fabricação possui o tipo de esquema de estados ilustrado na Figura 7.7. Pode-se observar nele os estados básicos "Pronto", "Suspenso", "Terminado" e "Completo". Novos estados (representados com fundo cinza no diagrama) também podem ser visualizados: "Pedido Registrado", "Produção Planejada", "Produção Finalizada", "Qualidade Testada" e "Entrega Confirmada". Esses novos estados especializam o estado básico "Executando". Segundo as regras definidas no Capítulo 4, há transições entre todos os estados básicos e os estados novos, correspondentes às transições entre esses estados básicos e o estado "Executando". Para simplificar a figura, foram omitidas as transições entre o estado Suspenso e os estados herdeiros de "Executando" (transições para retomar o serviço). O mesmo ocorrerá nos diagramas dos outros serviços. Nota-se que as atividades "Pedido Registrado", "Produção Planejada", "Produção Finalizada" e "Qualidade Testada" não possuem transições para o estado "Completo", somente para o estado "Terminado". Isso significa que o processo não pode ser considerado completo, isto é, finalizado com

sucesso, após atingirem-se esses estados. Somente após entrar no estado "Entrega Confirmada" o processo pode transitar para o estado "Completo". No entanto, a qualquer instante o serviço pode ser abortado pelo cliente, o que o faz ir para o estado "Terminado".

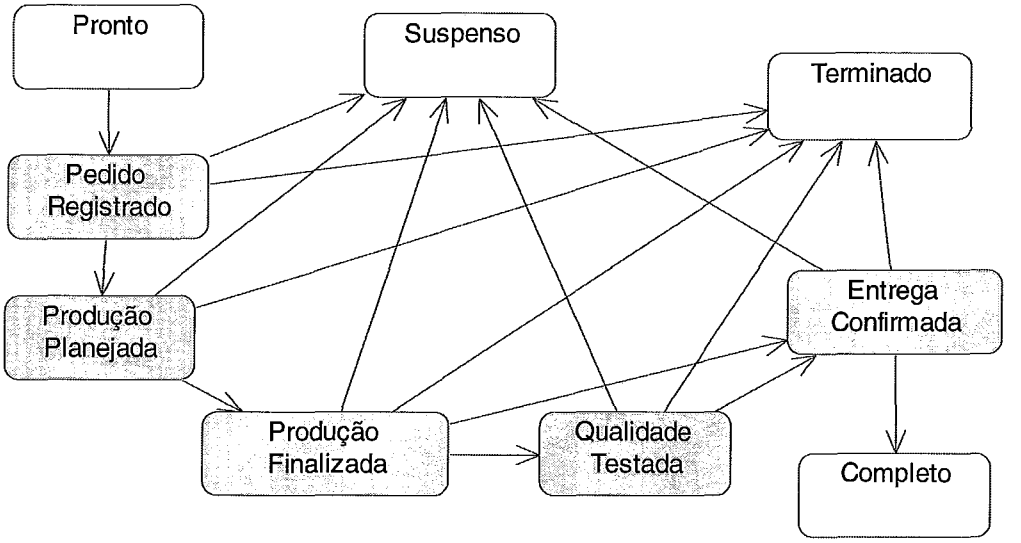

Figura 7.7 – Tipo de Esquema de Estados criado para o Serviço Fabricação

O tipo de esquema de estados ilustrado na Figura 7.8 é utilizado pelo serviço Entrega. Ele possui, além dos estados básicos, os seguintes novos estados: "Solicitação Registrada", "Planejamento Finalizado" e "Entrega Confirmada". Os dois primeiros possuem transições para o estado "Terminado" e não para "Completo", pelo mesmo motivo que ocorre no tipo criado para o serviço Fabricação. Porém, o estado "Entrega Confirmada" só possui transição para "Completo" e não para "Terminado". Isso acontece porque, uma vez entrado nesse estado, o serviço não pode mais ser abortado. Seu único caminho é transitar para "Completo", naturalmente, após as últimas ações.

Para o serviço Venda foi criado o tipo de esquema de estados ilustrado na Figura 7.9. Observam-se os seguintes novos estados: "Venda Ok", "Produto Encomendado", "Cliente Notificado", "Produto Configurado", "Entrega Solicitada" e "Satisfação Verificada".

Com se pode notar na Figura 7.9, após o produto ter sido encomendado, é possível passar para o estado "Produto Configurado" ou "Cliente Notificado". Somente depois de estar nesses dois estados é que o serviço pode entrar no estado "Entrega Solicitada" e seguir sua evolução. Após a satisfação do cliente ter sido verificada (o que denota o estado "Satisfação Verificada") o serviço pode entrar no estado "Completo". Como nos outros

serviços, antes disso, qualquer estado de execução pode levar ao estado "Terminado". Basta o cliente executar um comando de aborto/término forçado do serviço.

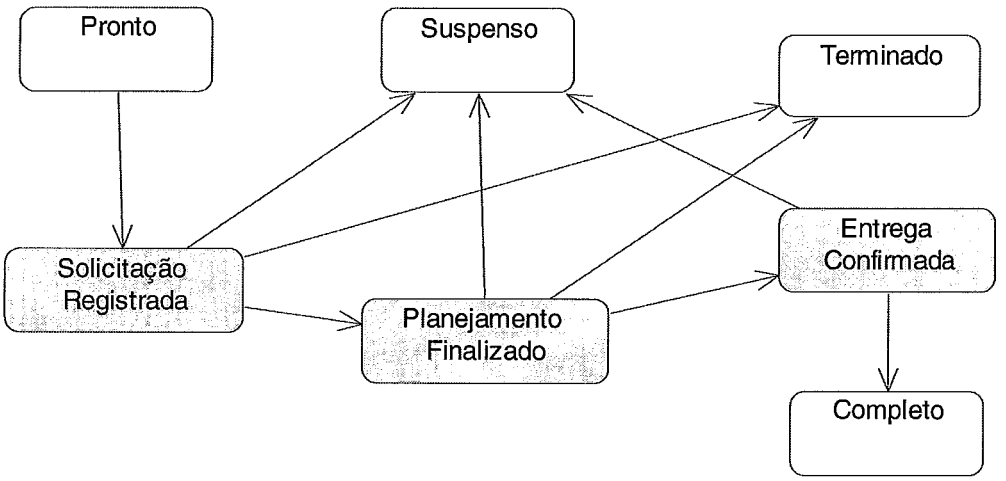

Figura 7.8 – Tipo de Esquema de Estados criado para o Serviço Entrega

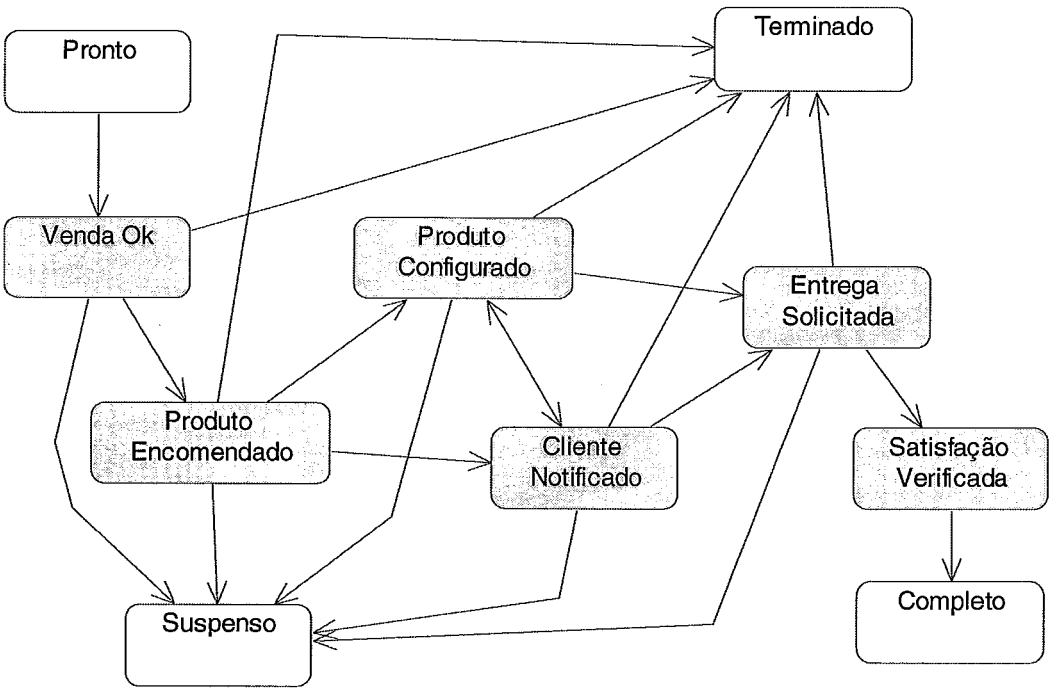

Figura 7.9 – Tipo de Esquema de Estados criado para o Serviço Venda

7.2.3. Processo de Fabricação de Roteadores

Uma determinada organização, digamos OrganizaçãoB, está interessada em fabricar produtos nesse processo virtual. Após serem realizados os acordos formais, essa organização passa a fazer parte da Organização Virtual. Ela implementa o serviço

Fabricação, mapeando-o para um de seus processos internos. As atividades desse processo podem ser vistas na Figura 7.10 e são descritas a seguir.

1) Registra pedido: confere empresa solicitante, registra pedido no BD, etc.
2) Verifica estoque: checa se há todos os componentes (matéria-prima) no estoque
3) Solicita fornecedor: se algum componente estiver em falta, é solicitado ao seus fornecedores (essa interação não foi modelada nesse exemplo)
4) Recebe componentes: recebe os componentes e verifica sua qualidade
5) Planeja produção: essa atividade, realizada pelo ERP da empresa, faz o planejamento da produção do produto solicitado
6) Produz: atividade de produção propriamente dita
7) Verifica qualidade: verifica a qualidade do produto produzido
8) Confirma entrega: confirma detalhes de entrega à empresa solicitante
9) Registra histórico da fabricação: registra um histórico do processo de fabricação, a fim de permitir sua posterior avaliação e melhoria

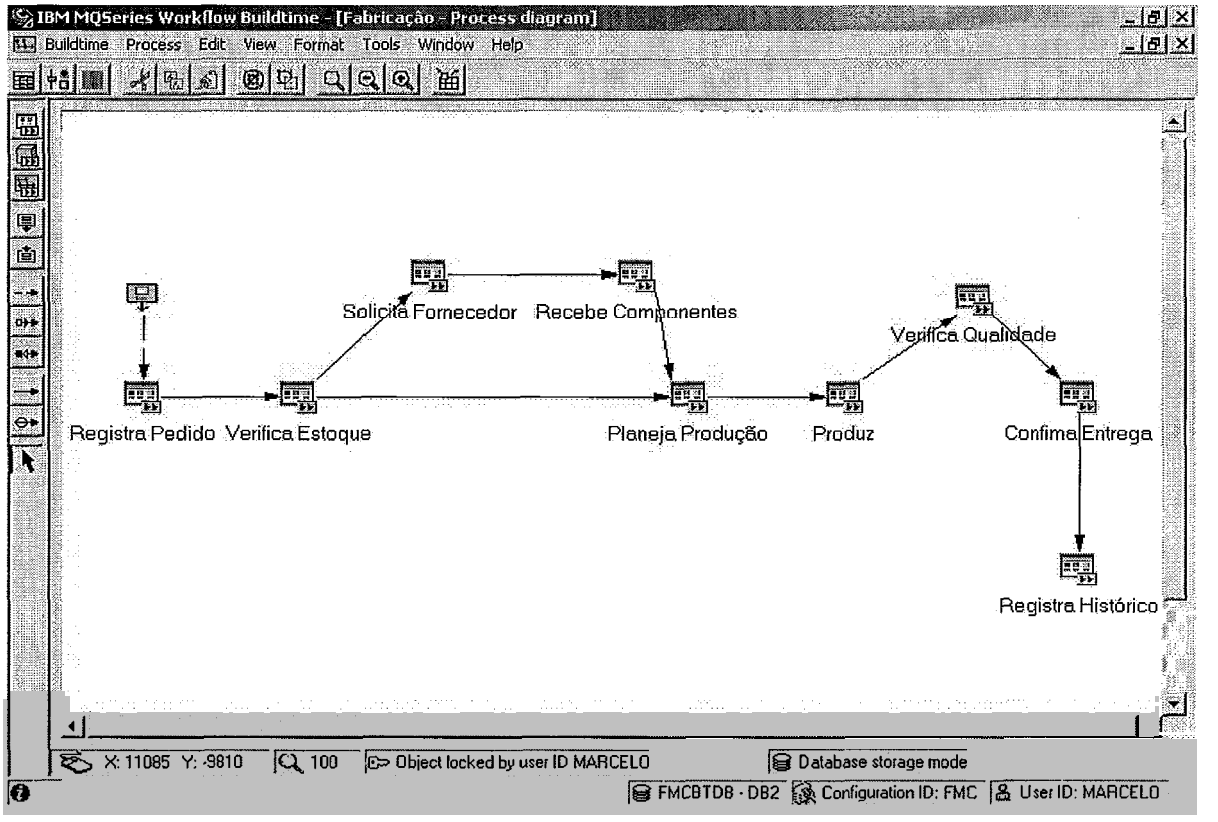

Figura 7.10 – Processo de Fabricação de Produto

Pode-se demonstrar, agora, como foi feito o mapeamento do processo interno da OrganizaçãoB para os estados do serviço de Fabricação (Tabela 7.1).

Estado	Condição
Pedido Registrado	Atividade 1 completa
Produção Planejada	Atividade 5 completa
Produção Finalizada	Atividade 6 completa
Produção Testada	Atividade 7 completa
Entrega Confirmada	Atividade 8 completa

Tabela 7.1 – Mapeamento de Estados para Atividades no Processo de Fabricação

A OrganizaçãoA, ao contratar os serviços de fabricantes, como a OrganizaçãoB, não precisa conhecer os detalhes de seus processos internos. Alterações nesses processos podem ser feitas, sem que isso afete a interação com as demais empresas.

A OrganizaçãoB, após a realização de um acordo com seus fornecedores que estabelece um prazo máximo para entrega dos componentes solicitados, resolver aprimorar seu processo de fabricação de roteador. O novo processo é apresentado na Figura 7.11.

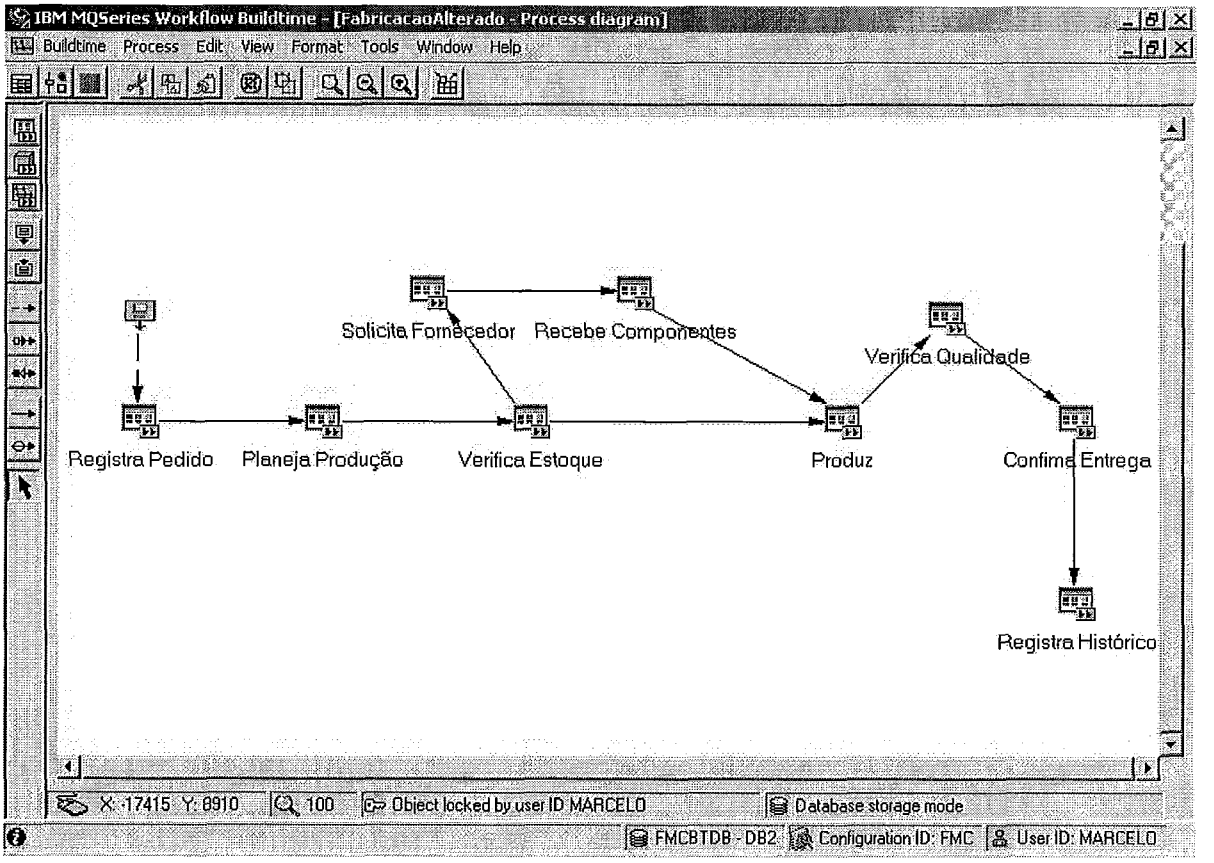

Figura 7.11 – Processo Alterado de Fabricação de Produto

Percebe-se, na Figura 7.11, que as atividades desse novo processo são as mesmas do processo anterior, porém houve uma mudança na sua coordenação. Com essa alteração, o fabricante consegue notificar antecipadamente seus clientes sobre o planejamento da produção. É importante enfatizar que, apesar do processo interno ter sido alterado, não houve quaisquer mudanças no serviço. Nesse caso, nem mesmo mudanças no mapeamento do serviço para os estados precisaram ser feitas.

7.2.4. Processo de Entrega

Uma terceira organização, OrganizaçãoC, passa a fazer parte da organização virtual ao decidir oferecer seus serviços de transporte para a entrega do produto. Após se estabelecer o contrato entre as partes, essa organização implementa o serviço Entrega através de seu processo interno.

O processo da transportadora representada pela OrganizaçãoC pode ser observado na Figura 7.12.

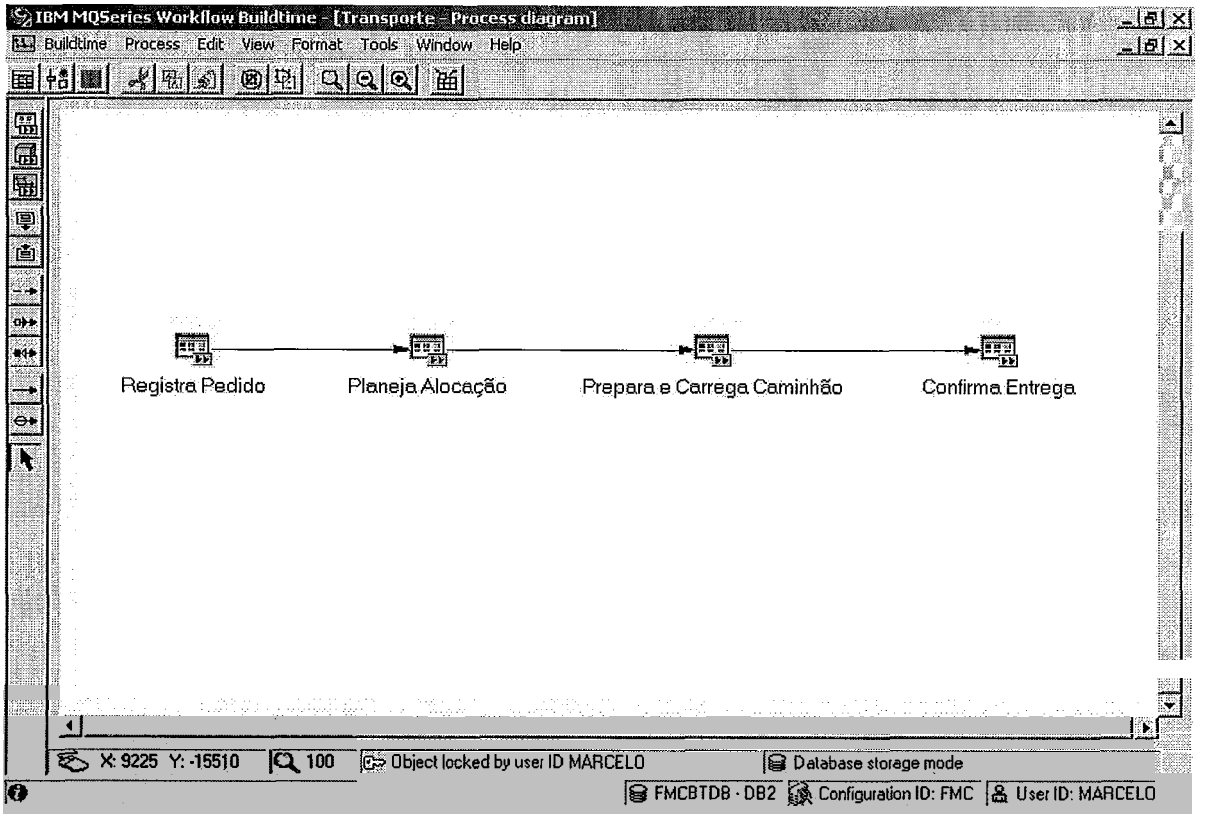

Figura 7.12 – Processo de Entrega

Esse processo é constituído pelas seguintes atividades:

1) Registra pedido: registra empresa solicitante, carga a ser entregue, etc.
2) Planeja alocação de caminhão: planeja a alocação de caminhões, de acordo com a carga, rota, entre outros parâmetros
3) Prepara e carrega caminhão: prepara o caminhão para o tipo de carga e o carrega
4) Confirma entrega ao cliente: indica à empresa cliente quando a carga chegará a seu destino

O mapeamento do serviço Entrega para o processo interno da OrganizaçãoC é mostrado na Tabela 7.2.

Estado	Condição
Solicitação Registrada	Atividade 1 completa
Planejamento Finalizado	Atividade 2 completa
Entrega Confirmada	Atividade 4 completa

Tabela 7.2 – Mapeamento de Estados para Atividades no Processo de Entrega

Uma possível alteração nesse processo seria confirmar a entrega logo após o planejamento. Novamente, isso não afetaria o serviço.

7.3. Conclusões Preliminares

Esse capítulo apresentou alguns aspectos da implementação da infra-estrutura, como a linguagem de programação empregada na codificação e os sistemas comerciais utilizados. Um desses sistemas, o MQSeries Workflow, foi descrito em mais detalhes, por ser o produto comercial mais importante utilizado na infra-estrutura.

Foi apresentado também um exemplo de desenvolvimento de aplicação de empresa virtual, utilizando-se a infra-estrutura. O exemplo escolhido foi inspirado num caso real. O processo virtual modelado conta com três atividades distintas. Como exemplo, os serviços que realizam essas atividades foram mapeados para três processos internos de três organizações diferentes.

Foi possível perceber algumas vantagens da integração de processos utilizando-se a infra-estrutura proposta. Primeiramente, detalhes técnicos como protocolos de comunicação, WfMS utilizado, linguagem de definição de processos, forma de invocação de operações, etc. foram abstraídos. Por utilizar o conceito de serviço para realizar as

interações, os processos puderam ser mantidos em sigilo, isto é, preservou-se sua privacidade. Algumas variações foram feitas sobre os processos, o que comprovou sua autonomia. Nota-se também que os processos não precisaram ser adaptados para serem integrados: continuaram independentes, como eram antes da integração. Apesar de só ter sido utilizada uma organização como exemplo de fornecedora para cada serviço, nada impediria que outras empresas fizessem parte da organização virtual, oferecendo serviços similares. Ou ainda, poder-se-iam utilizar serviços internos. A escolha da empresa adequada seria feita em tempo de execução, dinamicamente. A todo instante, foi possível monitorar e controlar os serviços sendo executados por outras organizações e, através de notificações, os serviços foram sincronizados (por exemplo, o processo de venda só verifica a satisfação do cliente após o recebimento de notificação do serviço Entrega indicando a data de entrega ao cliente). Por fim, pode-se observar que o processo virtual contém somente três atividades, relacionadas através de poucas interações, e que os processos internos foram mantidos intactos, o que caracteriza que não houve uma explosão de especificação, como ocorre em casos de integração de processos realizadas com os sistemas tradicionais.

Outras alterações poderiam ser feitas nos serviços e/ou processos internos, a fim de evoluir a empresa virtual. Uma possível mudança seria criar um novo serviço, especializado a partir do serviço Fabricação, para exigir que as empresas que o implementem realizem o processo de produção segundo as normas ISO 9000. O novo serviço seria chamado Fabricação ISO 9000 e teria alguns estados e comandos a mais, em relação ao serviço pai. No entanto, a atividade virtual que define o serviço Fabricação como seu realizador, no processo virtual, não precisaria ser alterada, uma vez que o novo serviço é uma especialização do serviço que ela já utiliza, seguindo o mesmo conceito da orientação a objetos.

Outra inovação possível seria oferecer o processo virtual de venda de roteador como um serviço para outras organizações. Por exemplo, uma organização poderia utilizar esse serviço para agregar valor a outro processo de venda, ainda mais amplo.

Todos os estados dos serviços foram mapeados utilizando-se condições simples, baseadas no estado "completo" das atividades. Porém, condições mais complexas poderiam

ter sido utilizadas, especificando-se outros estados de atividades, como "terminado", ou ainda composições destes (por exemplo, atividade 1 completa OU atividade 2 terminada).

8 - Conclusão

8.1. Introdução

Nesse trabalho, foram apresentados alguns padrões de *workflow* e comércio eletrônico. Padronizações são importantes no sentido de facilitar a integração entre produtos distintos, o que diminui o trabalho de integração de processos de negócios num nível mais amplo, entre empresas diversas, que é o foco dessa tese.

A seguir, um estudo dos problemas encontrados na questão de integração de processos entre empresas foi realizado. Foram levantados problemas referentes a aspectos tecnológicos, formas de definição de processos, autonomia, privacidade e independência de processos, problema relacionados à utilização de serviços e monitoramento, controle e sincronização de instâncias de processos. Foram mostrados também alguns sistemas, encontrados na literatura, que procuram resolver esses problemas.

Uma vez entendidos os problemas mais comuns que ocorrem ao se integrar processos entre empresas, passou-se então à elaboração dos conceitos e requisitos de um ambiente que permitisse o desenvolvimento de aplicações para gerenciamento de processos virtuais. Os principais conceitos propostos foram: serviço como mapeamento de processos internos, contratos, tipos de esquemas de estados (e sua forma de especialização), processos virtuais, utilização de sistemas comerciais de *workflow* e escolha dinâmica de serviços.

A partir desse conjunto de requisitos, foi planejada uma infra-estrutura que apoiasse a construção de tal ambiente. Seu modelo de dados e arquitetura foram detalhados e discutidos. O modelo de dados contém as seguintes categorias: Agente, Esquema de Estados, Serviço, Processo Virtual e Organização. Elas representam, respectivamente, agentes para a realização de serviços (aplicações e processos), diagramas de transições de estados, definições e instâncias de serviços, definições e instâncias de processos virtuais e, por fim, composição de organizações e contratos.

A arquitetura, distribuída, baseia-se no modelo RPC (Remote Procedure Call). Ela apresenta algumas ferramentas acessórias para definição de tipos e controle de execução de processos virtuais e uma máquina de execução. A máquina é composta por quatro

gerenciadores principais: Gerenciador de Serviços, Gerenciador de Contratos, Gerenciador de Processos Virtuais e Gerenciador Central. Para comunicação entre máquinas foi adotada a tecnologia de *web services* (protocolos SOAP e WSDL).

Posteriormente, o projeto da infra-estrutura foi apresentada, visando elucidar como foi planejada a construção de um ambiente com as características desejadas. Diversos *design patterns* foram utilizados para tornar o projeto mais flexível e extensível. Nessa parte do trabalho, mostrou-se como acontece o mapeamento de processos internos para serviços. Uma linguagem foi definida e implementada para o monitoramento de estados de atividades. Foi mostrado também como foram projetados os mecanismos que permitem efetuar o monitoramento e controle sobre serviços e processos virtuais.

Em seguida, foi discutida a implementação da infra-estrutura. Os componentes definidos no projeto foram construídos utilizando-se a linguagem Java e algumas tecnologias correlatas, como JAX-RPC (para comunicação através de *web services*) e JDBC (para acesso à base de dados). Alguns produtos comerciais também foram utilizados: IBM MQSeries Workflow como sistema de *workflow*, IBM MQSeries como gerenciador de comunicação (necessário ao funcionamento do *workflow*) e o SGBD IBM DB2.

Por último, um exemplo de aplicação, baseado num caso real, foi desenvolvido utilizando-se a infra-estrutura. O desenvolvimento e a execução desse exemplo serviram para validar a hipótese levantada no início desse trabalho e confirmar a eficácia da proposta elaborada.

8.2. Resultados Alcançados

Entre os resultados alcançados por essa tese, destaca-se a construção de uma infra-estrutura para apoio a processos virtuais. Essa infra-estrutura baseia-se no conceito de serviço como abstração de processos internos e as aplicações desenvolvidas nela, de fato, permitem o efetivo controle e monitoramento de serviços. Dos dez problemas citados, a infra-estrutura resolve total ou parcialmente oito deles: sistemas heterogêneos; definições de processos heterogêneas; privacidade de processos; autonomia e independência de processos; escolha dinâmica de serviços; explosão de especificação; monitoramento e controle de serviços; e sincronização de processos.

O projeto em si da infra-estrutura também se mostra como outra contribuição desse trabalho, uma vez que indica como se pode realizar a construção de um software de comércio eletrônico desse porte, baseando-se num projeto flexível, extensível e manutenível e fazendo uso de sistemas de *workflow*. Essa infra-estrutura baseou-se numa arquitetura distribuída, que utiliza técnicas de comunicação via www (*web services*), com ênfase no protocolo SOAP.

8.3. Dificuldades Encontradas

Do ponto de vista tecnológico, inicialmente havia-se planejado a utilização de ferramentas comerciais de modelagem para a definição de processos virtuais, com alguns artifícios para contemplar esse tipo de processo. Isto permitiria que, ao importar essa definição num WfMS, o macro-controle desses processos fosse feito automaticamente pelo sistema comercial. No entanto, por esse tipo de processo mostrar características bem peculiares de interação, isto não foi possível. Portanto, foi necessário oferecer recursos para a definição de processos virtuais e construir um gerenciador de processos virtuais, onde estes pudessem ser encenados.

Apesar da cultura de gerenciamento de processos e da tecnologia de *workflow* estarem em uso nas empresas já há alguns anos, a maioria delas ainda está empenhada na resolução da questão da encenação de seus processos internos, mesmo que a necessidade de integração com parceiros e clientes seja premente. Essa questão, aliada à natureza intrinsecamente complexa do problema, por lidar com diversas organizações ao mesmo tempo, tornou difícil a realização de um estudo de caso num contexto real.

8.4. Perspectivas e Trabalhos Futuros

Entre as possibilidades de trabalhos futuros está a construção de ferramentas gráficas para o desenvolvimento e execução de aplicações no ambiente. Ferramentas de definição de tipos, modelagem de processos virtuais e definição de serviços poderiam ser contempladas.

Uma ferramenta que facilitaria bastante o desenvolvimento de aplicações seria um compilador para a linguagem de expressão de condições de transições de estados. Através de uma ferramenta dessas, o usuário poderia definir as expressões textualmente e deixar a cargo do compilador transformá-la na árvore de objetos utilizada para representá-la e executá-la na infra-estrutura. Outra possibilidade seria definir um esquema XML (DTD ou XMLSchema) que permitisse que um usuário definisse de maneira mais simples uma árvore contendo a expressão de condição. Utilizando-se um *parser* XML, o arquivo XML contendo a expressão seria transformado na árvore de objetos correspondente.

Alguns problemas não foram solucionados pela infra-estrutura, como a descoberta e seleção automáticas de serviços e o controle de transação distribuído entre processos de empresas distintas. Procurar solucionar essas questões tornaria a infra-estrutura mais produtiva.

É preciso realizar experimentações num ambiente real para confirmar os resultados alcançados por esse trabalho. Com certeza, diversos novos problemas adjacentes serão descobertos, o que poderá dar origem a novas pesquisas nesse domínio.

Quando se trata da questão de integração de empresas, além do problema tecnológico (tratado nesse trabalho), existem diversos outros aspectos, como estabelecimento de contratos legais, cultura das empresas envolvidas e dos funcionários que nelas trabalham, etc. Certamente, esses aspectos também têm influência sobre o lado tecnológico da questão. Portanto, um estudo deles poderia levantar novos itens a serem pesquisados pelos profissionais interessados em prover a solução tecnológica para o problema.

9 - Referências

ALONSO, G., HAGEN C., SCHEK, H., J. et al., "Distributed Processing over Stand-alone Systems and Applications", In: *Proceedings of the 23rd International Conference on Very Large Databases (VLDB '97)*, Athens, Greece, Aug.

ALONSO G., FIEDLER, U., HAGEN A., et al., 1999, "WISE: Business to Business E-Commerce". In: *Proceedings of the 9th International Workshop on Research Issues on Data Engineering (RIDE-VE'99)*, Sydney, Australia, Mar.

ANT, 2002, *Apache Ant Project*, http://jakarta.apache.org/ant/index.html. Acesso mais recente: Março/2002.

APACHE, 2002, *Apache Software Foundation*, http://www.apache.org/. Acesso mais recente: Março/2002.

BAKER, D., GEORGAKOPOULOS, D., SCHUSTER, H. et al., 1999, "Providing Customized Process and Situation Awareness in the Collaboration Management Infrastructure". In: *Proceedings of the Fourth IFCIS Conference on Cooperative Information Systems (CoopIS '99)*, Scotland, Sep.

CAMPIONE M., WALRATH, K., HUML, A., 1999, *The Java Tutorial: A Short Course on the Basics*, 3ª ed., Sun Microsystems.

CASATI, F., 1998, A Discussion on Approaches to Handling Exceptions in Workflows. In: *Proceedings of CSCW '98, Towards Adaptive Workflow Systems Workshop.*

CASATI, F., DAYAL, U., SHAN, M., 2001, "Report on the VLDB Workshop on Technologies for E-Services (TES)", *ACM SIGMOD Record*, v. 29, n. 4.

COOPER, J., 1998, "Using Patterns Design", *Communications of ACM*, v. 41, pp. 65-68, Jun.

COPLIEN, J., 1992, *Advanced C++ Programming Styles and Idioms*, MA, EUA, Addison-Wesley, Reading.

DAMEN, Z., DERKS, W., DUITSHOF, M. et al., 2000, "Business-to-Business E-Commerce in a Logistics Domain". In: *Proceedings of the ISDO '00: The CaiSE*00 Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing*, Stockholm, Jun.

DAVENPORT, T. H., 1993, *Process Innovation, Reengineering Work through Information Technology*, Boston, Harvard Business School Press.

DAVIS, T. E., 1999, "Clever Facade makes JDBC look easy. Cool Tools", *JavaWorld*, Maio. Disponível em: http://www.javaworld.com/javaworld/jw-05-1999/jw-05-cooltools.html. Acesso mais recente: Março/2002.

DIAS, M. S., 1998, *COPSE: Um Ambiente de Suporte ao Projeto Cooperativo de Software*. Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

EBXML, 2001, *ebXML*, http://www.ebxml.org. Acesso mais recente: Março/2002.

EBXML-CPP, 2001, *Collaboration-Protocol Profile and Agreement Specification*, Version 1.0.

EBXML-TA, 2001, *Technical Architecture Specification*, Version 1.0.4.

EDIFACT, 2002, *UN/EDIFACT Standard Directory*, http://www.unece.org/trade/untdid/welcome.htm. Acesso mais recente: Março/2002.

FURLAN, 1998, J. D., *Modelagem de Objetos através da UML*, 1ª ed., São Paulo, Makron Books.

GAMMA E., HELM, R., JOHNSON, R et al., 1995, *Design Patterns – Elements of Reusable Object-Oriented Software*, MA, EUA, Addison-Wesley.

GEORGAKOPOULOS, D., SCHUSTER H., CICHOCKI, A. et al., 1999a, "Managing Process and Service Fusion in Virtual Enterprises", *Information Systems*, v. 24, n. 6, pp. 429-456.

GEORGAKOPOULOS, D., SCHUSTER H., RUSINKIEWICZ, M., 1999b, "Collaboration Processes Management for Advanced Application", Relatório Técnico do MCC.

GREFEN, P., ABERER, K., HOFFNER, Y et al., 2000, *Cross-Flow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises*, Relatório Técnico do projeto CrossFlow.

HAMMER M. e CHAMPY J., 1993, *Reengineering the Corporation: A Manifesto for Business Revolution*, New York, HarperCollins.

HAN, Y., SHETH, A., BUSSLER, C., 1998, "A Taxonomy of Adaptive Workflow Management". In: *Proceedings of CSCW '98, Towards Adaptive Workflow Systems Workshop.*

IBM-DB2, *DB2 Home-Page*, http://www-4.ibm.com/software/data/db2/. Acesso mais recente: Março/2002.

IBM-MQWF, 2001a, *IBM MQSeries Workflow Version 3.3, Programming Guide*, 7ª ed.

IBM-MQWF, 2001b, *IBM MQSeries Workflow Version 3.3, Getting Started with Buildtime*, 7ª ed.

IBM-MQWF, 2001c, *IBM MQSeries Workflow Version 3.3,Concepts and Architecture*, 4ª ed.

IBM-WS, 2002, *IBM Web Services ToolKit – A showcase for emerging web services technologies.*

JACOBSON I., CHRISTRERSON, M., JONSSON P. et al., 1992, *Object-Oriented Software Engineering – A Use Case Driven Approach*, 1ª ed., Addison-Wesley.

JACOBSON I., ERICSSON, M. e JACOBSON, A., 1995, *The Object Advantage – Business Process Reengineering with Object Technology*, 1 ed., ACM Press.

JAKARTA, *The Jakarta Project*, http://jakarta.apache.org/. Acesso mais recente: Março/2002

JAX-RPC, 2002, *Java(TM) API For XML-Based RPC (JAX-RPC)*, http://java.sun.com/xml/jaxrpc/index.html. Acesso mais recente: Março/2002.

JDBC, 2001, *JDBC Data Access API*, http://java.sun.com/products/jdbc/. Acesso mais recente: Março/2002.

KLINGEMANN, J., 2000, "Controlled Flexibility in Workflow Management". In: *Proceedings of the 12th International* Conference on Advanced Information Systems Engineering (CAiSE'00), Stockholm, Sweden, Jun., pp. 126-141.

KLINGEMANN J., WÄSCH, J., ABERER, K., 1999, "Adaptive Outsourcing in Cross-Organizational Workflows". In: *Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99)*, Heidelberg, Germany, Jun., pp. 417-421.

LAZCANO, A., ALONSO, G., SCHULDT, H. et al., 2000, "The WISE Approach to Electronic Commerce", *International Journal of Computer Systems Science & Engineering, special issue on Flexible Workflow Technology Driving the Networked Economy*, v. 15, n. 5, Sep.

LAZCANO, A., SCHULDT, H., ALONSO, G. et al., 2001, "WISE: Process Based E-Commerce", *IEEE Data Engineering Bulletin, Special Issue on Infrastructure for Advanced E-Services*, v. 24, n.1.

LAZCANO, A., ALONSO, G., 2001, "Process Based E-Services". In: *Proceedings of the Second International Workshop on Electronic Commerce (WELCOM'01)*, Heidelberg, Germany, Nov.

LUDWIG, H., HOFFNER, Y., 1999, "Contract-based Cross Organizational Workflows – The CrossFlow Project". In: *Proceedings of the WACC Workshop on Cross-Organizational Workflow Management and Co-Ordination.*

LUDWIG, H., BUSSLER, C., SHAN, M. et al., 1999, *Cross-Organizational Workflow Management and Co-ordination WACC '99 Workshop Report.*

MARGHERIO, L., HENRY, D., COOKE, S. et al., 1995, *The Emerging Digital Economy*, U.S. Department of Commerce.

NAUR, P. (ed.), 1963, "Revised Report on the Algorithmic Language Algol-60", *Numerische Mathematik,* 4, pp 420-453.

ODBC, 1999, *Open Database Connectivity (ODBC)*, Microsoft Corporation, http://www.microsoft.com/data/odbc. Acesso mais recente: Março/2002.

OMG, 1998, *The Common Object Request Broker: Architecture and Specification*, Revision 2.2.

OMG-TS, 1996, *Object Management Group and X/Open, RFP5 Submission: CORBA Trading Object Service*, Document orbos/96-05-6.

OMG-WMF, 2000, *Workflow Management Facility Specification, OMG*, Version 1.2.

ROSETTANET, 2002, *RosettaNet Home,* http://www.rosettanet.org. Acesso mais recente: Março/2002.

RUMBAUGH J., BLAHA M., PREMERLANI, W. et al., 1994, *Modelagem e Projetos baseados em Objetos*, 1ª ed., Rio de Janeiro, Editora Campus.

SCHMID, B. F., 1997, "Requirements for Electronic Markets Architecture", *EM - Electronic Markets*, V. 7, N. 1 (Feb.).

SWENSON, K., 1998, *Simple Workflow Access Protocol (SWAP)*, IETF, Draft Proposal.

TOMCAT, 2002, *Jakarta Tomcat*, http://jakarta.apache.org/tomcat/index.html. Acesso mais recente: Março/2002.

UDDI, 2000, *UDDI Technical White Paper, Universal Description, Discovery and Integration Organization (uddi.org).*

VINCENT, A. F., 2002, "Uma Proposta de Método para explicitar Processos em Sistemas de Informação Existentes", Tese de M.Sc., IM-NCE/UFRJ, Rio de Janeiro, RJ, Brasil.

VONK, J., GREFEN, P., BOERTJES E. et al., 1999, "Distributed Global Transaction Support for Workflow Management Applications". In: *Proceedings of the 10th International Conference on Database and Expert Systems Applications*, Florença, Itália.

W3C-SOAP, 2000, *Simple Object Access Protocol (SOAP) 1.1 W3C Note*, http://www.w3.org/TR/SOAP. Acesso mais recente: Março/2002.

W3C-WSDL, 2001, *Web Services Description Language (WSDL) 1.1 W3C Note*, http://www.w3.org/TR/wsdl.html. Acesso mais recente: Março/2002.

W3C-XML, 2000, *Extensible Markup Language (XML) 1.0 (Second Edition)*, http://www.w3.org/TR/REC-xml. Acesso mais recente: Março/2002.

WFMC-TC00-1003, 1995, *The Workflow Reference Model, Workflow Management Coalition*, Document Number TC00-1003, Issue 1.1.

WFMC-TC-1009, 1998, *Workflow Management Application Programming Interface (Interface 2&3) Specification, Workflow Management Coalition*, Document Number WfMC-TC-1009, Version 2.0.

WFMC-TC-1011, 1999, *Terminology & Glossary, Workflow Management Coalition*, Document Number WfMC-TC-1011, Version 3.0.

WFMC-TC-1012, 1999, *Interoperability Abstract Specification, Workflow Management Coalition*, Document Number WfMC-TC-1012, Version 2.0b (Draft).

WFMC-TC-1016, 1999, *Process Definition Interchange, Process Model, Workflow Management Coalition*, Document Number WfMC-TC-1016, Version 1.1.

WFMC-TC-1018, 2000, *Interoperability Internet e-mail MIME Binding, Workflow Management Coalition*, Document Number WfMC-TC-1018, Version 1.2.

WFMC-TC-1023, 2001, *Workflow Standard - Interoperability, Wf-XML Binding, Workflow Management Coalition*, Document Number WfMC-TC-1023, Version 1.1.