UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Protótipo de um Sistema de Gerenciamento de Workflows Baseado em Eventos

Autor: **Benito Tomás Giordani**

Orientador: **Prof. Dr. Manuel de Jesus Mendes**

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Campinas - SP Brasil
1999

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G437p

Giordani, Benito Tomás
Protótipo de um sistema de gerenciamento de workflows baseado em eventos. / Benito Tomás Giordani.--Campinas, SP: [s.n.], 1999.

Orientador: Manuel de Jesus Mendes
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Reengenharia de sistemas. 2. Engenharia de produção – Automação. 3. Tecnologia da informação. 4. Redes de computação. I. Mendes, Manuel de Jesus. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Protótipo de um Sistema de Gerenciamento de Workflows Baseado em Eventos

Benito Tomás Giordani

Dissertação de Mestrado defendida e aprovada, em 03 de maio de 1999, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Manuel de Jesus Mendes, Presidente
PUCCAMP/UNICAMP

Prof. Dr. Fernando Antonio Campos Gomide
UNICAMP

Prof. Dr. Ralph Santos da Silva
CTI

Resumo

Este trabalho descreve uma implementação de um sistema de gerenciamento de workflow baseado nos conceitos de computação orientada a eventos e objetos distribuídos. O protótipo foi construído pela fusão de versões simplificadas do modelo proposto pela jFlow[12], adotado pela OMG (Object Management Group), do modelo de tarefas proposto na especificação da Nortel[13] e do modelo de workflow genérico, apresentado no modelo de referência da WfMC[6], em um único e coerente sistema de gerenciamento de workflow. O modelo de coordenação das atividades está baseado no modelo de tarefa proposto pela Nortel e as interfaces das entidades consideradas, que constituem o sistema de gerenciamento, estão consistentes com a especificação da jFlow. Baseado no modelo de tarefa da Nortel, este trabalho também apresenta uma proposta que contém os elementos básicos de uma linguagem de definição de processo. Todos os componentes foram implementados como objetos CORBA[2].

Abstract

This work describes an implementation of a Workflow Management System. The implementation is based on the concepts of event-driven computing and distributed objects. The software prototype was built by combining simplified versions of the jFlow model [12], adopted by the OMG (Object Management Group), the task model proposed by Nortel[13] and the Workflow Reference Model of the WfMC[6] into a single and coherent workflow management system. The activities coordination model is based after the task model presented by Nortel. The interfaces of the included entities of the Workflow Management System are consistent with the jFlow Specification. After the Nortel task model this work also presents a proposal that includes the basic elements of a process definition language. The components were implemented as CORBA[2] objects.

Agradecimentos

Ao meu pai Antonio Giordani, á minha mãe Maria José Marangoni Giordani, aos meus irmãos Antonio Frederico, Enrico José e Endrigo Emanuel Giordani, aos meus sobrinhos Ana Carolina Pedro Giordani e Antonio Frederico Giordani Júnior, ao meu tio Antonio Galvão Marangoni, à minha avó Dona Ernesta Palini Marangoni pelo carinho, compreensão e incentivo durante todos os momentos da minha vida.

Ao meu orientador Manuel de Jesus Mendes pela confiança, apoio, conselhos e liberdade dada para desenvolver o meu trabalho. Também agradeço o professor Waldomiro Loyolla pelo auxílio durante as fase iniciais do trabalho.

A todos os meus ex-professores da graduação, em particular a professora Maria Cristina Fraga e o professor Juan Coello pela educação por eles me dada. Também a todos os professores da pós-graduação.

Aos meus amigos Marcelo Malheiros, Marcelo Tilli, Lizet, Luiz Ramirez, José Guerrero, Affonso Guedes, Gonzaga, Faina, Alexandre, Rodrigo Prado, Miglinsky, Raquel Cristina, Jane Marian, Viviane Petito, Matheus de Paiva, Cristina Trielli, Cleane de Oliveira, Sandro, Lorenzo, Juares, Mariana e a todos aqueles que neste momento o nome não me vêm a memória, pela ajuda de forma direta e indireta na conclusão deste trabalho.

Neste parágrafo quero fazer uma homenagem ao meu grande amigo e professor José Soto Mejia. Esta que é a pessoa que durante o desenvolver deste trabalho me auxiliou no entendimento e fixação dos respectivos conceitos teóricos. Também fico grato por ter compartilhado comigo e as demais pessoas com quem se relaciona as discussões filosóficas, conselhos e experiências próprias no que diz respeito aos padrões de relacionamentos interpessoais aos quais todas as pessoas convivem hoje em dia. Caro colega espero que os seus ensinamento acompanhem-me pelo resto da vida e que me auxiliem a encarar melhor os problemas que irei me deparar. Espero que a nossa amizade também assim prossiga. Um grande abraço do seu amigo Benito Tomás Giordani.

Financiador

Agradeço à Capes pelo apoio financeiro.

Índice

Lista de Figuras

Capítulo 1 *Introdução*

A tendência mundial em direção a um mercado globalizado tem exacerbado a competitividade das empresas, tornando muito mais críticos, e importantes, alguns aspectos gerenciais tais como a qualidade, custo, confiabilidade, flexibilidade, rapidez focalizando os processos organizacionais.

Um dos principais elementos em desenvolvimento tendo como alvo a automação e o gerenciamento automatizado de processos de negócio têm sido os Sistemas de Gerenciamento de Workflows.

Um workflow pode ser definido como o fluxo de informação e controle em um processo de negócio. Um sistema de gerenciamento de workflow (workflow management system) é um sistema que permite criar definições de processos e gerenciar a execução de workflows [6]. Basicamente é composto de um programa que interpreta definições de processo e delega a execução das tarefas, especificadas nesta definição, às entidades executoras existentes no ambiente de negócio. Sistemas de gerenciamento de base de dados (SGBD) e recursos IT (Information Technology - Tecnologia de Informação), são exemplos de entidades executoras automáticas, enquanto que pessoas ou grupos de pessoas interagindo com programas de aplicação são exemplos de entidades executoras semi-automáticas.

A reengenharia do processo de negócio é uma disciplina que tem por objetivo a otimização dos processos de negócio. Ela conduz inevitavelmente à reengenharia dos sistemas de informação e, em particular, dos sistemas de gerenciamento de workflows. Devido aos possíveis traumas causados pelas atualizações dos sistemas IT (seja pelo alto custo ou pela instabilidade que a atualização possa trazer) as empresas estão relutantes em fazer mudanças nos seus sistemas de informação.

Portanto, é necessário uma arquitetura que permita rápida alteração dos sistemas de informação para responder às mudanças nos processos de negócio. Deste ponto de vista,

um passo para alcançar este objetivo está na componentização dos sistemas de informação e, em particular, dos sistemas de gerenciamento de workflows.

Os modelos atuais de sistemas de gerenciamento de workflows diferenciam-se pela ênfase que dão a determinadas funcionalidades de tal sistema. O modelo de referência da Workflow Management Coalition [6] foi a primeira e, atualmente, a mais completa tentativa feita para descrever e padronizar os elementos constituintes de um sistema de gerenciamento de workflows. Apesar de todo o esforço realizado, o WfMC-RM [6] não é considerado um padrão *de facto*. Outras empresas que já vinham desenvolvendo pesquisas nesta área, também se dispuseram a participar desse conflito.

No intuito de solucioná-lo, a OMG (Object Management Group) divulgou um documento[10] convidando as empresas, que desenvolvem produtos e pesquisa na área de workflow, a enviarem propostas de modelos que cumpram (não necessariamente todos) os requisitos contidos neste documento.

Dentre os diversos modelos submetidos, os dois que mais resistiram foram os modelos da jFlow[12] e Nortel[13]. O modelo da jFlow [12], adotado pela OMG como a "Workflow Management Facility" de sua arquitetura, dá ênfase à interoperabilidade entre os componentes que participam no gerenciamento de workflows durante o momento de execução do processo. O modelo da Nortel [13], embora vise a interoperabilidade, também define um modelo de coordenação de tarefas baseado em eventos.

Este trabalho se propõe a: (i) verificar a factibilidade do mapeamento e a fusão de versões simplificadas do modelo da jFlow [12] adotado pela OMG, do modelo de tarefa proposto na especificação da Nortel [13] e do modelo de sistema de gerenciamento de workflows genérico apresentado no modelo de referência da WfMC [6], em um único e coerente sistema de gerenciamento de workflows; (ii) criar um protótipo simples, porém funcional, de um sistema de gerenciamento de workflows que permita servir como base para experimentos futuros; (iii) apresentar os elementos básicos de uma linguagem de definição de processos coerente com o modelo de tarefa da Nortel [13]; (iv) manter na implementação a separação dos aspectos do modelo de workflow que são mais prováveis de mudar com mais freqüência, daqueles aspectos que tendem a permanecer constantes.

No Capítulo 2 são apresentados os conceitos gerais de workflow e sistema de gerenciamento de workflows. No Capítulo 3 é descrito o funcionamento de um sistema de gerenciamento de workflows segundo a visão da WfMC, o modelo, submetido a OMG, da jFlow e da Nortel. No Capítulo 4 encontra-se o protótipo do sistema de gerenciamento e a linguagem de definição de processos propostos. No Capítulo 5 encontra-se a descrição da modelagem de um processo, existente no mundo real, com a linguagem de definição de processos proposta, descrição da simulação deste processo

pelo sistema de gerenciamento de workflow construído e alguns aspectos da implementação do protótipo. No Capítulo 6 descrevem-se as conclusões e propostas de trabalhos futuros. No Apêndice A está especificado o conjunto de interfaces IDL que definem as entidades criadas e no Apêndice C há um resumo das notações UML mais utilizadas neste trabalho.

Capítulo 2

Workflows

2.1. Conceitos Gerais

O conceito de workflow evoluiu da noção de processo utilizado nas fábricas e escritórios. Tais processos existem desde o início da industrialização e são produto da busca para o aumento da eficiência, concentrando na rotina os aspectos das atividades do trabalho [15].

Não há uma única definição do que vem a ser workflow e quais as características que um WFMS (Workflow Management System - Sistema de Gerenciamento de Workflow) deve prover. Para aumentar a eficiência, de maneira geral, as atividades do trabalho são separadas em tarefas, papeis (responsabilidades (roles)), regras e procedimentos que regulam a maioria do trabalho nos escritórios e nas fábricas [15]. O primeiro vestígio histórico de especificação de workflow apareceu com a modelagem de escritório (office modeling), que era uma maneira de descrever os procedimentos dentro deste ambiente. Tais descrições eram baseadas em extensões dos modelos formais clássicos, tais como as redes de Petri, regras de produção e fluxogramas.

Com a introdução da tecnologia de informação (IT - Information Tecnology), os processos do ambiente de trabalho foram parcial ou totalmente automatizados. Entende-se por tecnologia de informação toda a infra-estrutura tecnológica, e.g., computadores, sistemas gerenciadores de base de dados (SGBD), sistemas operacionais de rede, etc, que é utilizada para facilitar a automatização dos processos de negócio.

Medina-Mora et al. [26] classificam os processos de uma organização em três categorias a saber: (i) processos materiais, (ii) processos de informação, (iii) e processos de negócio.

O escopo dos processos materiais envolve montar e manusear componentes físicos, tais como: mover, armazenar, transformar, medir e contar objetos do mundo real.

Exemplos de processos materiais são, entre outros, a linha de montagem de uma máquina qualquer (carro, computador, brinquedo, etc) e o processo de refinamento de minério.

Os processos de informação estão relacionados às tarefas automatizadas (i.e., tarefas executadas por programas de computador) ou parcialmente automatizadas (i.e., tarefas executadas por pessoas interagindo com computadores). Estas tarefas criam, processam e fornecem informações aos seus usuários. Geralmente, os processos de informação estão relacionados à estrutura organizacional e/ou ao ambiente do sistema de informação existente. Bases de dados, monitores de processamento de transação e tecnologias para o processamento distribuído compõem a infra-estrutura básica para suportar os processos de informação.

Processos de negócio são descrições, orientadas ao mercado, das atividades de uma organização, implementados como processos de informação e/ou materiais. Um processo de negócio é implementado para servir a um contrato ou satisfazer uma necessidade específica do consumidor. Assim, a noção de processo de negócio, conceitualmente, está num nível mais elevado que a noção de processo material ou de informação [15]. A venda em lote de um produto, a criação de um novo produto ou serviço, etc, são exemplos de processos de negócio.

Uma vez que uma organização captura o seu negócio em termos de processos de negócio, pode-se fazer a reengenharia, em cada processo, para melhorá-lo ou adaptá-lo às mudanças de necessidades. Algumas razões para se reprojetar um processo de negócio incluem: (i) aumentar a satisfação do consumidor; (ii) melhorar a eficiência das operações do negócio; (iii) aumentar a qualidade do produto; (iv) reduzir custos; (v) enfrentar novos desafios; (vi) oportunidades mudando-se os serviços existentes ou introduzindo-se novos. A reengenharia do processo de negócio (BPR – Business Process Reenginering) envolve a re-avaliação e o reprojetar dos processos de negócio. A reengenharia dos processos de informação é uma atividade complementar à reengenharia de processos de negócio. Nela determina-se como usar os sistemas de informação legados ou novos para otimizar tais processos existentes no ambiente. A reengenharia de processo de negócio explicitamente envolve a satisfação do consumidor; a reengenharia de processos de informação envolve a eficiência e custo dos sistemas de informação e pode tirar proveito dos avanços tecnológicos existentes.

O conceito de workflow está muito próximo aos conceitos de reengenharia e automação de processos de negócio e de informação nas organizações. Um workflow pode descrever as tarefas de um processo de negócio em um nível conceitual necessário para entendê-lo, avaliá-lo e reprojetá-lo. Por outro lado, um workflow pode capturar as tarefas de um processo de informação em um nível conceitual que descreva as exigências do processo em termos de funcionalidades do sistema e habilidades humanas. A distinção

entre estas perspectivas nem sempre é feita e, algumas vezes, o termo workflow é usado para descrever tanto a perspectiva de negócio quanto a do sistema de informação [15].

Schulze [30] diz que o processo de negócio existe no mundo real independentemente do conceito de workflow. Ele denomina esquema de workflow ou descrição de workflow ao subconjunto do processo de negócio que pode ser modelado utilizando-se os elementos de modelagem de um metamodelo de workflow.

A WfMC [6] (Workflow Management Coalition) define workflow como sendo a facilitação computacional ou automatização do processo de negócio, no todo ou em parte. Geogakopaolus [15] define-o como uma coleção organizada de tarefas para se realizar um processo de negócio qualquer. Amith Sheth e Rusikiesk [23] definem workflow como sendo as atividades que envolvem a execução coordenada das diversas tarefas, desempenhadas por diferentes entidades processadoras, dentro de um ambiente. Uma tarefa define algum trabalho a ser feito e pode ser especificada de várias maneiras, tais como uma descrição textual em um arquivo email, um formulário, ou um programa de computador. A entidade processadora que desempenha a tarefa pode ser uma pessoa ou um software (e.g., um mailer, um programa de aplicação, ou um sistema de gerenciamento de base de dados). É interessante notar a neutralidade tecnológica nas duas últimas definições, tornando as definições mais genéricas e atraentes.

A especificação de workflow envolve a descrição das características das atividades que o compõem (e as entidades processadoras que as executam), e que sejam relevantes para controlar e coordenar a execução do processo de negócio. A especificação de workflow é também denominada por alguns autores de modelo de processo, *template* de processo, metadado de processo, definição de processo, esquema de workflow, *script* de workflow, etc [6]. Esta especificação também deve conter os relacionamentos entre as tarefas, i.e., dependências e exigências de execução.

O gerenciamento de workflow (WFM) é o suporte tecnológico à reengenharia de processos e de informação. Ele envolve: (i) definir o workflow, i.e., descrever os aspectos de um processo que são relevantes para coordenar e controlar a execução de suas tarefas (e possivelmente as habilidades dos indivíduos ou sistemas de informação exigidos para desempenhar cada tarefa) e (ii) prover meios para o rápido (re)projeto e (re)implementação dos processos, à medida que as necessidades do negócio e do sistema de informação mudarem [15].

Para suportar o WFM, as organizações devem evoluir o seu ambiente de sistema de informação para um novo ambiente distribuído que (i) seja orientado a componentes, i.e., suporte a integração e interoperabilidade entre componentes com baixo acoplamento, que na realidade serão os sistemas novos ou legados com características heterogêneas,

autônomas e/ou distribuídas (HAD - Heterogeneous, Autonomous and Distributed); (ii) suporte aplicações workflow, i.e, implementações dos processos de negócio ou de informação, que acessam múltiplos sistemas HAD; (iii) garanta a confiabilidade e precisão dos dados das aplicações na presença de concorrência e falha (dependable systems); e (iv) suporte a evolução, substituição e adição de aplicações workflow e componentes de sistema, a maneira que os processos sofram a reengenharia [15].

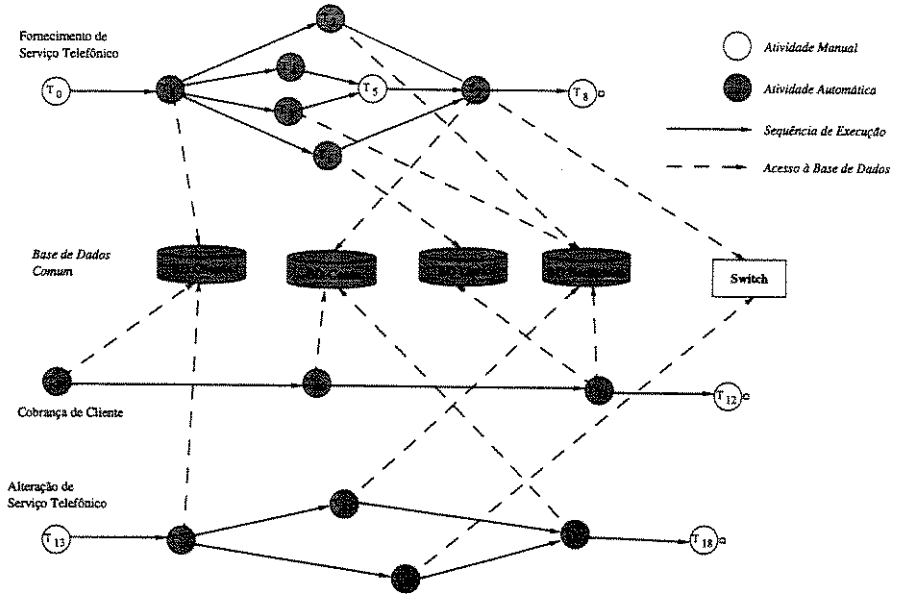

Figura 2.1 - Exemplo de um workflow em telecomunicações [15]

Freqüentemente o workflow está associado ao conceito de BPR (Business Process Re-Engineering - Reengenharia de Processo de Negócio), que se preocupa com a avaliação, análise, modelagem, definição e subsequente implementação operacional do processo de negócio chave de uma organização (ou outra entidade de negócio). Embora nem todas as atividades de BPR resultem na implementação de workflows, esta tecnologia freqüentemente é uma solução apropriada, pois suporta a separação da lógica procedimental do negócio de seu suporte operacional em termos de IT, possibilitando que subsequentes mudanças sejam incorporadas às regras que definem o processo de negócio. De maneira análoga, nem todas as implementações de workflow necessariamente formam parte do exercício de BPR [6].

2.1.1. Classificação de workflows

Geogakopaolus [15] apresenta, em seu artigo, duas categorizações de workflows: (i) uma proposta pelos fabricantes de produtos workflows e, (ii) outra proposta por ele mesmo.

2.1.1.1. Classificação pelos fabricantes de produtos workflow

Os fabricantes de sistemas gerenciadores de workflow (Workflow Management System - WFMS) caracterizam os workflows em três tipos: *ad hoc*, administrativo e de produção. As dimensões nas quais estes tipos de workflows são descritos incluem: (i) previsibilidade e repetitibilidade do processo; (ii) como o workflow é iniciado e controlado, e.g., semi- ou automatizado; (iii) aspectos relativos à precisão dos dados manipulados por tais sistemas.

Workflows ad hoc são aqueles onde não há um padrão permanente de movimentação de informação entre as entidades processadoras. A dependência entre as tarefas muda constantemente e fica muito complexo criar uma especificação de workflows para cada caso que poderia ocorrer. Além do mais, nada se conhece a respeito da duração das tarefas que compõem este tipo de workflow. As tarefas de um workflow *ad hoc* envolvem a coordenação, colaboração e/ou co-decisão entre pessoas. Assim, a coordenação e o controle das tarefas neste tipo de sistema não são automatizadas, mas controladas por pessoas e feitas em tempo de execução do workflow. Envolvem grupos pequenos de pessoas e geralmente dão suporte às atividades de curta duração que exigem soluções rápidas.

Os WFMSs *ad hoc* não são utilizados em ambiente de missão crítica, i.e., falhas periódicas não interferem significativamente no processo de negócio como um todo. A infra-estrutura tecnológica atualmente utilizada em WFMSs *ad hoc* varia de servidores de email a sistemas de agendamento e de conferência para grupos. Usualmente usam base de dados proprietária para armazenar informações compartilhadas (e.g., documentos como os formulários de pauta da conferência ou artigos). Os WFMSs que suportam workflows *ad hoc* são denominados de *groupware* [15].

Workflows administrativos são aqueles caracterizados pela repetitibilidade e previsibilidade dos processos. Estes, normalmente, possuem regras simples de coordenação de tarefas, tais como o roteamento de relatório de custos ou de viagem num pedido de permissão. A coordenação e o controle das tarefas podem ser automatizados. Os WFMSs que suportam workflows administrativos manipulam funções de roteamento de informação e aprovação de documentos, por exemplo, requisições de compra ou planejamento de viagens. Os workflows administrativos não englobam processos de informação complexos e não necessitam acessar múltiplos sistemas de informação. Geralmente não são de missão crítica. A infra-estrutura tecnológica atualmente utilizada está baseada em correio eletrônico [15].

Workflows de produção envolvem processos de negócio repetitivos e previsíveis, tais como as aplicações de empréstimo bancário ou acionamento de seguro. Ao contrário

dos workflows administrativos, os workflows de produção envolvem tipicamente processos de informação complexos que precisam fazer o acesso a múltiplos sistemas de informação. O controle e a coordenação de tais workflows podem ser automatizados. Entretanto a automação de workflows de produção é complicada devido a: (i) complexidade do processo de informação; e (ii) acessos a múltiplos sistemas de informação para desempenharem o trabalho e recuperarem dados para a tomada de decisão (workflows administrativos baseiam-se nas pessoas para a maioria das decisões e execução do trabalho). Os WFMSs que suportam o workflow de produção devem prover facilidades para definir as dependências entre as tarefas e controlar a execução delas com pouca ou nenhuma intervenção humana. Freqüentemente são de missão crítica e devem lidar com a integração e interoperabilidade dos sistemas de informação HAD [15].

As diferenças entre o workflow de produção, do workflow ad hoc ou administrativo são: (i) a interação dos sistemas de informação com o processo de negócio; (ii) o uso de executores de tarefas automáticos. Veja na figura 2.2 a relação entre o tipo de workflow e a complexidade da tarefa.

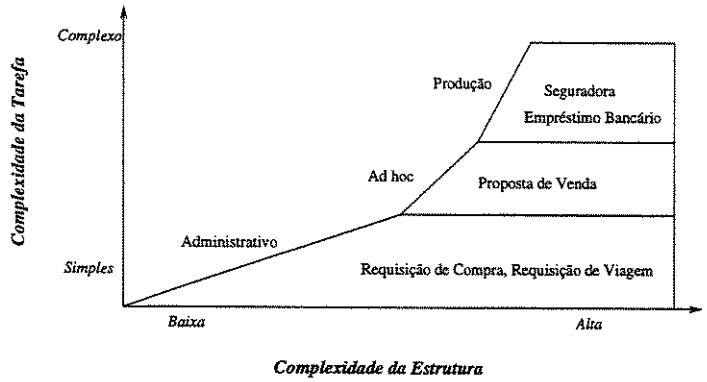

Figura 2.2 - Relação entre o tipo de workflow e a complexidade da tarefa [15]

2.1.1.2. Classificação segundo Georgakopoulos

Georgakopoulos [15] caracteriza os workflows sobre uma linha contínua, sendo que, de um dos extremos desta, estão os workflows orientados a seres humanos e do outro, os orientados a sistemas (figura 2.3).

O workflow orientado a humanos enfoca a colaboração das pessoas na execução e coordenação das tarefas. As exigências do sistema para este ambiente são para o suporte a coordenação e colaboração entre as pessoas, objetivando com isso a melhoria da vazão do trabalho. Entretanto, tais sistemas de gerenciamento de workflow não garantem nenhuma consistência dos documentos e dos resultados produzidos, deixando esta tarefa ser feita pelas pessoas.

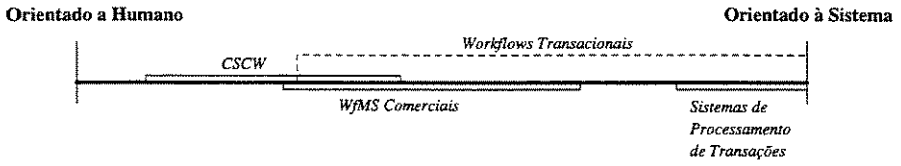

Figura 2.3 - Caracterização de workflow segundo Georgakopoulos [15]

No outro extremo, os workflows orientados a sistemas envolvem sistemas de computadores que desempenham operações de computação intensiva e tarefas de software especializadas, além de acessarem sistemas de informação HAD.

Enquanto que as implementações de workflow orientados a humanos controlam e coordenam as tarefas humanas, as implementações de workflow orientados a sistemas controlam e coordenam as tarefas de software, geralmente, com pouca ou nenhuma intervenção humana. As implementações de workflow orientados a sistemas devem incluir software para o controle de concorrência e técnicas de recuperação de falhas para garantir a consistência e confiabilidade dos dados. Isto não é exigido e não pode ser provido pelos WFMSs que suportam workflows orientados a humanos. Estes sistemas possuem a semântica dos processos mas não têm conhecimento da semântica das informações que estão sendo processadas. Desta maneira, o WFMS existe para auxiliar as pessoas e não será o responsável pela manutenção da consistência dos dados.

Em workflows orientados a humanos, os principais pontos a serem considerados são: (i) interação homem-máquina; (ii) conciliação entre as habilidades humanas com as exigências das tarefas; (iii) facilitação para a customização de acordo com a cultura do ambiente, i.e., como as pessoas precisam ou preferem trabalhar.

Em workflows orientados a sistemas, os principais pontos que devem ser considerados são: (i) garantir a interoperabilidade com e entre os sistemas HAD; (ii) encontrar a tarefa de software apropriada para desempenhar a tarefa de workflow; (iii) determinar novos softwares necessários para automatizar o processo de negócio; (iv) garantir a execução correta e confiável do sistema.

Tópicos como tratamento de exceções, priorização e prazo de término (*deadline*) podem aparecer em diferentes formas em ambos os tipos de sistema e devem ser abordados. Na figura 2.3, são mostrados segmentos que indicam a faixa das características de workflows que aparecem em sistemas: (i) CSCW (Computer Suported Cooperative Work); (ii) WFMS comerciais; (iii) sistema de processamento de transações (TP) comerciais (e.g., DBMSs distribuídos, monitores de TP). Nota-se que a área de CSCW compartilha características com alguns WFMS comerciais onde o workflow envolve predominantemente tarefas humanas. Sistemas de TP comerciais compartilham

características com o WFMS quando as aplicações de workflow são submetidas como transações DBMS ou de monitores de TP.

Workflows transacionais são aqueles que coordenam e executam tarefas que (i) podem envolver pessoas; (ii) exigem acessos a sistemas HAD; e (iii) necessitam de suporte ao uso seletivo de propriedades transacionais (ACID - Atomicity, Concurrency, Intependency and Durability - Atomicidade, Consistência, Isolação e Durabilidade) para tarefas específicas ou no processo como um todo. O uso seletivo de propriedades transacionais é exigido para permitir funcionalidades especializadas inerentes aos workflows (e.g., permitir a colaboração de tarefas e suporte a estruturas de workflow complexas (aninhadas)). O modelo transacional dos DBMSs e monitores de TP não permite o uso seletivo de propriedades transacionais para funcionalidades especializadas (e.g., as transações providas por DBMS impõem isolação e isto não permite a cooperação das tarefas ou workflows), e por isso é necessário torná-los flexíveis ou extendê-los para suportar as exigências funcionais do workflow. Segundo Georgakopoulos [15], os workflows de hoje em dia não suportam aspectos-chave de um workflow orientado a sistema.

2.2. Gerenciamento de Workflow

O gerenciamento de workflow envolve todos os aspectos, desde a definição de processo até à sincronização das tarefas, executadas pelos sistemas de informação e pessoas, pertencentes ao ambiente do processo do negócio [15]. Em particular, o gerenciamento de workflow inclui os seguintes passos (figura 2.4): (i) modelagem de processo e especificação de workflows: modelos de workflows e metodologias para capturar os processos como especificações de workflow; (ii) reengenharia de processo: metodologias para otimizar os processos; e (iii) automação de workflows: metodologias e tecnologias que façam uso de sistemas de informação e pessoas para implementar, escalonar, executar e controlar as tarefas do workflow como descrito na sua especificação. Este último item é o que chamamos de sistema de gerenciamento de workflow (WFMS). O passos citados acima são descritos mais detalhadamente abaixo.

Deste ponto em diante, o termo processo também é utilizado para denominar processo de negócio.

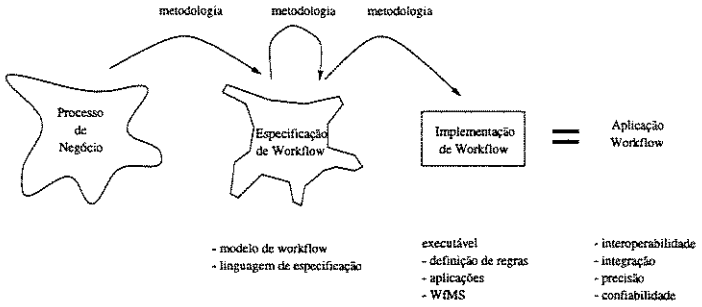

Figura 2.4 - Passos para a automação de workflows [15]

2.2.1. Modelagem de Processo e Definição de Workflow

A definição de processo é um modelo resultante da atividade de modelagem de processo. É utilizado para auxiliar no entendimento e reengenharia do processo e pode servir de base para a criação da definição de workflow.

Tanto a definição de processo, quanto a definição de workflow, especificam, entre outros aspectos, as tarefas que constituem o processo, a seqüência e as condições de ativação delas, as características da entidade responsável em executá-las. Porém o que diferencia estes dois conceitos é o nível de detalhe contido e a utilidade do modelo [15].

Segundo Georgakopoulos [15] o termo "definição de processo" é mais adequado para denominar um modelo que especifica as peculiaridades do processo, acima descritas, em um alto nível conceitual (independente de detalhes tecnológicos) muito utilizado por reengenheiros para compreender, avaliar e possivelmente re-modelar o processo visando melhorar, entre outras coisas, o desempenho deste último. O termo "definição de workflow" é mais adequado para definir um modelo em um baixo nível conceitual mostrando, entre outros aspectos, detalhes relativos aos recursos humanos e/ou IT utilizado pelos sistema de gerenciamento de workflow durante a execução do processo (mais adiante encontra-se a explicação do que vem a ser um sistema de gerenciamento de workflows). Segundo ele, a definição de workflow é criada com base na definição de processo.

Para se criar uma definição de processo normalmente são usadas técnicas de entrevista, como as utilizadas por analistas de sistemas para capturar as exigências do sistema. Metodologias como os "casos de uso" [18] e a análise essencial [28] são adequadas nesta fase. Recentemente a UML (Unified Modeling Language) formalizou um conjunto de notações para especificar processos de negócio [8]. Após esta primeira atividade, toda a informação adquirida deve ser registrada em uma definição de processo

A definição de workflow geralmente é feita usando-se uma linguagem de definição de workflow. As linguagens de definição de workflow, utilizadas comercialmente hoje

em dia, basicamente permitem o uso de regras, limitações (constraints), e/ou símbolos gráficos, para especificar a ordem, a sincronização, os atributos e os papéis das tarefas. Na seção 2.3.3. é mostrado como a definição de workflow se insere no contexto da automação de workflows.

Geogakopoulos [15] identifica duas linhas de metodologias que apoiam a modelagem de processos: (i) orientadas à comunicação, e (ii) orientadas a atividades.

As *metodologias orientadas à comunicação* derivam do trabalho desenvolvido por Winograd e Flores [29]. Nessa metodologia assume-se que o objetivo da reengenharia de processo é aumentar a satisfação do consumidor, sendo o objetivo modelar o compromisso entre as pessoas. A modelagem nesta metodologia consiste em reduzir todas as atividades de um processo a quatro fases baseadas na comunicação entre o consumidor (cliente) e o produtor (executor) (figura 2.5): (i) preparação - um consumidor requisita uma ação a ser desempenhada ou o produtor se oferece a produzir alguma coisa; (ii) negociação - o produtor e o consumidor entram em acordo sobre a ação a ser executada e definem os termos de satisfação; (iii) execução - a ação é executada nos termos estabelecidos no acordo; (iv) confirmação - o consumidor relata a satisfação (ou insatisfação) com a ação desempenhada.

Cada *loop* entre o produtor e o consumidor pode ser composto de outros *loops* para completar um processo de negócio. O produtor num *loop* pode ser o consumidor noutro. A especificação resultante revela uma rede social, na qual um grupo de pessoas, atuando sob vários papéis, completa um processo de negócio. A falta de suporte a tarefas paralelas é uma desvantagem desta metodologia.

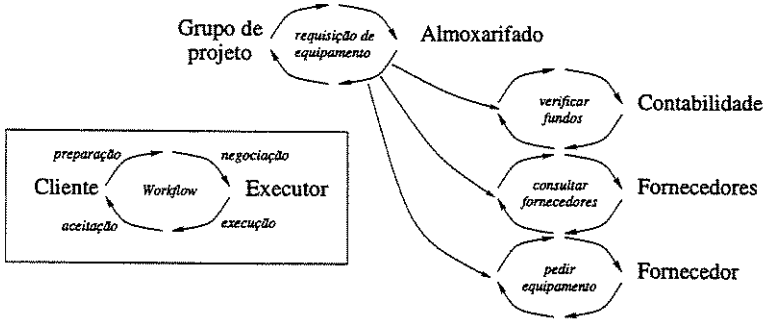

Figura 2.5 - Modelo orientado a comunicação [15]

Metodologias orientadas a atividades enfocam a modelagem do trabalho em si, e têm como objetivo definir a natureza do processo, as tarefas que o compõem, seus atributos, as dependências entre elas e os papéis necessários para executá-las. Os esforços feitos para a padronização das linguagens de especificação de workflow, geralmente, adotam esta linha metodológica.

Os modelos de workflow baseados em atividade ou comunicação podem ser combinados quando os objetivos do processo de negócio são compatíveis com os objetivos de ambos os modelos (e.g., satisfazer o cliente minimizando-se as tarefas no workflow e papéis das pessoas).

Metodologias orientadas a objetos, como as propostas por Rumbaugh [20] e Jacobson [18] podem ser úteis na especificação de processos de negócio (e para derivar implementações). Por exemplo, Jackobson descreve como (i) identificar objetos que correspondam a "atores" i.e., papéis no workflow; (ii) identificar dependência entre estes objetos; (iii) usar conceitos de orientação a objetos, tais como, herança para organizar a especificação; e (iv) descrever "casos de uso", que essencialmente são uma seqüência de tarefas necessárias para completar um processo de negócio.

A UML possui extensão para fazer a modelagem de processos de negócio [8]. Esta extensão é composta por um conjunto de símbolos e significados que apoiam o conceito de processos de negócio.

A modelagem de processo de negócio pode ser feita independentemente do uso da tecnologia de workflow. As metodologias acima citadas são de grande utilidade para a reengenharia de processos, pois permitem ao engenheiro de processos analisar, modelar, registrar e alterar os processos. Na próxima seção aborda-se como o modelo de processo pode ser útil para um sistema de gerenciamento de workflows.

2.2.2. Reengenharia de Processo

As metodologias de reengenharia de processos têm como objetivo otimizar o processo de negócio. As estratégias de otimização de processos dependem dos objetivos da reengenharia (e.g., aumento da satisfação do consumidor, redução dos custos dos negócios, introdução de novos produtos e/ou serviços). As metodologias de reengenharia atualmente são uma arte. A especificação de processos provê uma descrição em alto nível de um processo que facilite o questionamento de sua eficiência.

2.2.3. Automação de Workflows

A automação de workflows envolve os aspectos referentes à utilização de recursos humanos (pessoas ou grupo de pessoas) e IT (computadores, softwares, sistemas de informação, e/ou sistemas de gerenciamento de workflows) para executar um workflow.

Os Sistemas de Gerenciamento de Workflows (WFMS - Workflow Management System) são aqueles que provêem a automação do workflow, gerenciando a seqüência das atividades e a invocação dos recursos humanos e/ou IT (Information Tecnology -

Tecnologia de Informação) necessários, que foram associados com as diversas atividades que compõem o processo [6].

Há duas abordagens para a automação de workflows que diferenciam-se pela flexibilidade propiciada pelo sistema de gerenciamento de workflows: (i) *hardcoded*; (ii) *softcoded*. Na abordagem *hardcoded*, a definição de processo e/ou a definição de workflow são utilizadas pelos projetistas de software como uma espécie de documento de requisitos para construirem um sistema de gerenciamento de workflows especializado para o processo em questão. Na abordagem *softcoded* (que foi utilizada para construir o protótipo), os projetistas criam um sistema de gerenciamento de workflows desacoplado de uma definição de processo em particular, porém com a capacidade de interpretar a linguagem de definição de processo utilizada para construí-la. Isto permite maior flexibilidade na alteração do comportamento do sistema de gerenciamento de workflows, durante a execução do processo, apenas alterando a definição de workflow utilizada. (A partir deste ponto, sempre que houver referência a um sistema de gerenciamento de workflows faz-se referência a um sistema de gerenciamento de workflows construído sobre a abordagem *softcoded*).

A maioria dos WFMSs dão suporte a automação de apenas um subconjunto do processo de negócio, i.e., existem determinadas partes do processo de negócio que não se conseguem automatizar pela sua complexidade e imprevisibilidade. A figura 2.6 mostra uma abstração para ilustrar este cenário. Note que a definição de processo (linha tracejada) não consegue especificar todos os detalhes de um processo de negócio e a perda é maior ainda quando se cria a definição de workflow (linha pontilhada).

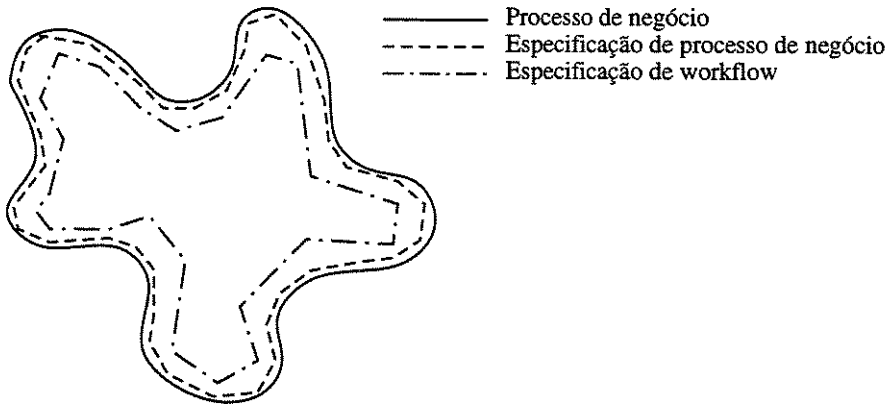

Figura 2.6 - Perda de informação devido a fatores tecnológicos

Para que um processo de negócio seja automatizado, através do uso de um WFMS, primeiro ele deve ser modelado e descrito numa forma computacionalmente executável. Nesta fase algumas informações podem ser perdidas pela falta de suporte dos WFMSs.

Apesar de haver diferenças entre definição de processo e especificação de workflow, daqui para frente estes termos são utilizados indistintamente para referir-se a esta notação computacionalmente executável. Outros termos encontrados na literatura são: (i) esquema de workflow, (ii) modelo de workflow, etc.

A partir do momento em que os dados iniciais e a definição de processo já tenham sido definidos, o sistema de gerenciamento de workflows interpreta esta definição e delega a execução das atividades especificadas aos recursos necessários. No próximo capítulo, estão detalhados os principais componentes que compõem um sistema de gerenciamento de workflows.

Capítulo 3

Sistema de Gerenciamento de Workflows

Neste capítulo são descritos os principais componentes e o funcionamento de um sistema de gerenciamento de workflow segundo o modelo da WfMC. Também são apresentados dos modelo de sistema de gerenciamento de workflows propostas à OMG.

3.1. Workflow Management Coalition

Nesta seção descreve-se o modelo do sistema de gerenciamento de workflows sobre o ponto de vista da WfMC com o objetivo de auxiliar no entendimento de tal sistema e o modelo de referência proposto por esta organização.

A Workflow Management Coalition (WFMC), fundada em 1993, é uma organização internacional, sem fins lucrativos, de desenvolvedores, usuários, analistas, universidades e grupos de pesquisa na área de workflow. A missão da WfMC é promover e desenvolver o uso de workflows, estabelecendo padrões para a terminologia, interoperabilidade e conectividade entre produtos de workflow [6]. Esta organização publica diversos documentos que especificam, entre outras, as interfaces entre os principais componentes de um sistema de gerenciamento de workflows [6].

A figura 3.1 mostra os principais componentes, identificados pela função que desempenham, de um sistema de gerenciamento de workflows segundo a visão da WfMC ou seja, função de projeto, função de tempo de execução e função de interação. A figura 3.2 apresenta maiores detalhes de tal sistema e o foi o modelo utilizado para se construir o modelo de primeiro nível do protótipo (Capítulo 4). A seguir segue a descrição dos componentes da figura 3.1 e seus detalhes.

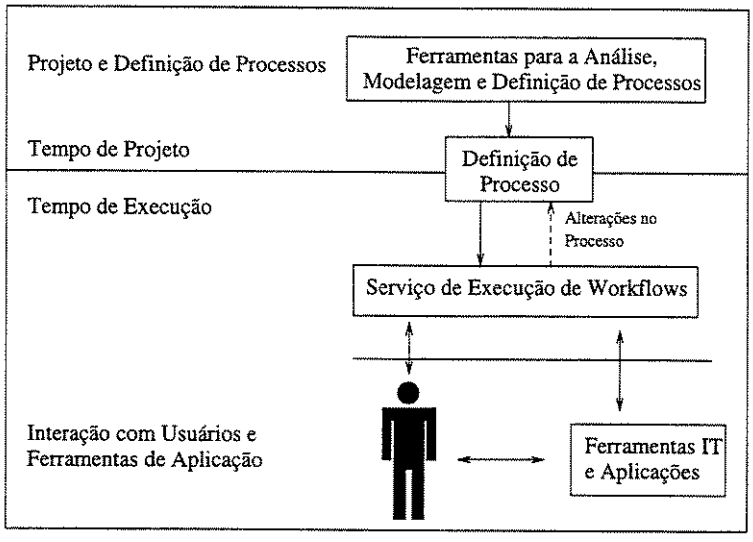

Figura 3.1 - Principais componentes de um sistema de gerenciamento de workflows [6]

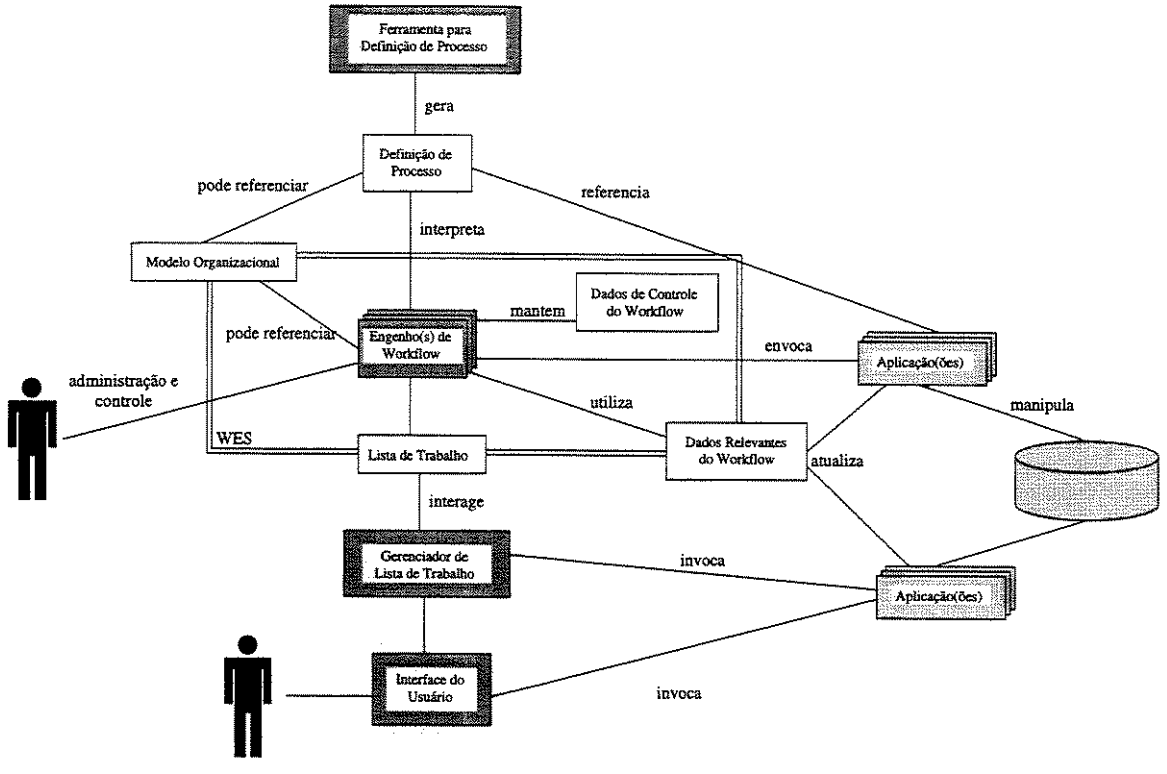

Figura 3.2 - Estrutura geral de um sistema de gerenciamento de workflows [6]

3.1.1. Funções em Tempo de Projeto

Para que um determinado processo de negócio seja automatizado com o uso de um sistema de gerenciamento de workflows, primeiro deve-se transformar sua lógica em uma representação processável por tal sistema. Esta representação é denominada de definição de processos (PD - Process Definition). Os aplicativos que dão suporte à construção da definição de processo são chamados de Ferramentas para a Definição de Processos (PDT - Process Definition Tools).

As PDTs são usadas para criar a definição de processo. As PDTs atuais possuem interface gráfica para facilitar a criação e visualização do processo de negócio como um todo. A definição pode ser baseada numa linguagem formal de definição de processos, num modelo relacional de objetos ou, caso seja um sistema simples, num *script* ou num conjunto de comandos de roteamento para transferir informações entre as atividades. A definição de processo gerada por uma PDT deve ser compatível com a linguagem esperada pela máquina de workflows. Quando a compatibilidade não for direta, deve-se adotar algum mecanismo para o mapeamento entre as duas. Às vezes, no mapeamento pode ocorrer perda de informação, sendo que, na maioria dos casos, isto é inadmissível. Uma PDT pode ser suprida como parte de um WFMS ou pode ser parte de um produto independente e específico para a análise de processo de negócio, que possua outros componentes para a análise, modelagem e simulação de processos de negócio.

A PD é o resultado da modelagem do processo de negócio. Contém toda a informação necessária sobre o processo para permitir sua execução pelo WFMS, mais precisamente pela máquina de workflows [6]. São informações sobre as condições de iniciação e término do processo, as atividades que o compõem e regras para a navegação entre estas, além das tarefas que devem ser executadas pelos usuários, referências a aplicações que podem ser invocadas e definição de qualquer elemento de dado que o WFMS referencie para tomar suas decisões.

Há duas maneiras de associar entidades executoras a uma determinada tarefa durante a criação da definição de processos, quais sejam: (i) associação direta, (ii) associação indireta [16].

A associação direta é caracterizada pela especificação na definição de processos do nome da entidade executora responsável em executar as tarefas que constituem este processo.

A associação indireta é caracterizada pelo uso de um modelo auxiliar chamado de modelo organização-papel que basicamente mantém uma lista de pares nome de papel e entidade executora que possui a capacidade de se colocar neste papel. Isto permite especificar na definição de processos qual o papel necessário que a entidade executora deve ter para executar uma dada tarefa. Desta maneira, a associação entre a tarefa e a entidade executora responsável em realizar o trabalho desta atividade é feita dinamicamente no momento de execução do processo pelo sistema de gerenciamento de workflows.

Por exemplo, imagine os seguintes cenários: (i) a atividade A deve ser executada pela entidade executora I, que possui o papel P dentro da organização; (ii) a atividade A deve ser executada por qualquer entidade executora, que desempenhe o papel P dentro da organização. A definição de processo do cenário (ii) permite maior reusabilidade da PD em caso de mudança no ambiente organizacional. As entidades executoras de um ambiente empresarial mudam com o decorrer do tempo, i.e., pessoas são demitidas ou mudam de cargos (papéis) ou, softwares são atualizados ou substituídos.

Alguns grupos têm despendido muito esforço para estabelecer padrões para as linguagens de definição de processos, permitindo a comunicação não somente entre ferramentas de definição de processo e WFMSs, mas também a troca de informações entre ferramentas de análise de processo. Entre os mais conhecidos podemos citar (i) a WPDL (Workflow Process Definition Language) que foi concebida pela WfMC para permitir a interoperabilidade entre ferramentas de definição de processo e sistemas gerenciadores de workflow [4]; (ii) o PIF (Process Interchange Format) [19], linguagem resultante do projeto PIF; (iii) e a PSL (Process Specification Language) [5] também uma linguagem resultante de um projeto com o mesmo nome.

3.1.2. Funções em Tempo de Execução

Em tempo de execução a definição de processo será interpretada por um software que é responsável pelas operações de criação e controle das instâncias dos processos, escalonando as diversas atividades dentro dele e delegando o trabalho de execução destas tarefas aos recursos humanos e sistemas IT necessários. O componente núcleo é o software de controle do gerenciamento básico de workflow, responsável pela criação, exclusão, controle do escalonamento das atividades dentro de um processo operacional e pela interação com as ferramentas de aplicação e recursos humanos, que são utilizados para executar as tarefas. Este software é freqüentemente distribuído em diversas plataformas de computação, pois as atividades em um processo de negócio são de natureza distribuida.

O Serviço de Execução de Workflows (*Workflow Enactment Service - WES*) é o componente que controla a instanciação de processos. Cada instância de processo é executada por um ou mais máquinas de workflows (workflow engine) cooperativas, que gerenciam a(s) execução(ões) das instâncias individuais dos diversos processos. O WES mantém dados de controle interno, tanto centralizados quanto distribuídos, entre um conjunto de máquinas de workflows. Os dados de controle de workflow (workflow control data) incluem informações do estado interno associado às diversas instâncias de processo e atividades que estão em execução. Também podem incluir pontos de verificação e informações de recuperação/reinicialização usados pelas máquinas de workflows para voltarem a um estado consistente sob situações de falha.

A definição de processo, junto com qualquer dado relevante ao workflow (workflow relevant data) é usada para controlar a navegação, i.e., escalonamento das atividades, provendo informações

sobre os critérios de entrada e saída, opções de execução paralela ou seqüencial e sobre as tarefas dos usuários ou sistemas IT associadas a elas. Durante este processo, pode-se necessitar de acesso a dados do modelo organização/papel, caso a definição de processo inclua construções relacionadas com este modelo.

Existem duas vertentes para a construção de uma máquina de workflows que estão relacionadas à linguagem de definição de processo: (i) controle do processo centralizado, e (ii) controle do processo distribuído [25].

Em uma máquina de workflows com o controle centralizado os dados processados pelas entidades executoras retornam a um elemento central (máquina de workflows) que determina a seqüência de atividades a ser executada.

Sobre o paradigma de controle distribuído, a máquina de workflows interpreta a definição de processo e, conforme especificado, configura as atividades para reagirem a um determinado conjunto de eventos. Desta maneira, as tarefas que compõem o processo de negócio conhecem suas dependências (tanto temporais como de dados) e iniciam sua execução, quando estas dependências forem satisfeitas. Este modelo ficará mais claro quando for apresentado o modelo de tarefas, especificado na proposta da Nortel [13].

As duas entidades que se seguem provêem funções de controle em tempo de execução e também funções de interação.

Os *Dados das Aplicações* são os dados a serem processados ou produzidos pelas aplicações. Os *Dados Relevantes ao Workflow* são todos os dados utilizados pela máquina de workflows para determinar a seqüência de ativação das tarefas.

A *Lista de Trabalho* (*Worklist*) é uma entidade que gerencia um conjunto de atividades que estão prontas para serem executadas. Esta lista basicamente é utilizada para associar executores às tarefas não automatizadas ou semi-automatizadas (necessitam de interação humana). Se uma dada tarefa estiver pronta para ser executada ela é colocada em uma lista (worklist) que as entidades executora semi-automáticas (pessoas ou grupo de pessoas) podem acessar. As tarefas podem ser atribuídas a estas entidades executoras de duas maneira: (i) modelo *pop* - é responsabilidade do entidade verificar as tarefas que estão esperando a execução e executar qualquer uma delas; (ii) modelo *push* - a worklist envia uma mensagem à entidade informando-lhe qual atividade ela deve executar.

3.1.3. Funções de Interação

As tarefas dentro de um processo são executadas por entidades executoras automáticas ou semi-automáticas. Assim é necessário algum tipo de interação entre a entidade que está controlando a execução do processo e as entidades que podem executar estas atividades.

A interação com o software de controle de processos é necessária para transferir o controle entre as atividades, para certificar o estado operacional dos processos, para invocar as ferramentas de aplicação e para passar os dados apropriados. Há vários benefícios em se padronizar a estrutura para suportar este tipo de interação, incluindo-se o uso de interface consistente para múltiplos sistemas workflow e a capacidade de desenvolvimento de ferramentas de aplicação comuns, que trabalhem com diferentes produtos de workflow.

A WfMC foi a primeira tentativa de se estabelecer padrões para a interoperabilidade em sistemas de workflows. Dentro das atividades desempenhadas por este órgão estão: (i) a padronização de APIs (Application Programing Interface) para acesso aos serviços/funções de um WFMS; (ii) especificação de formatos e protocolos entre WFMSs, e entre WFMSs e aplicações; (iii) especificação de formatos para a permuta de modelos de especificações de workflow entre múltiplos WFMSs. Recentemente foi publicado um documento sobre a segurança em sistemas de gerenciamento de workflows [14] e a especificação de interfaces para permitir interoperabilidade entre WFMSs e sistemas e aplicações IT. A figura 3.3 mostra as principais interfaces que a WfMC pretende padronizar.

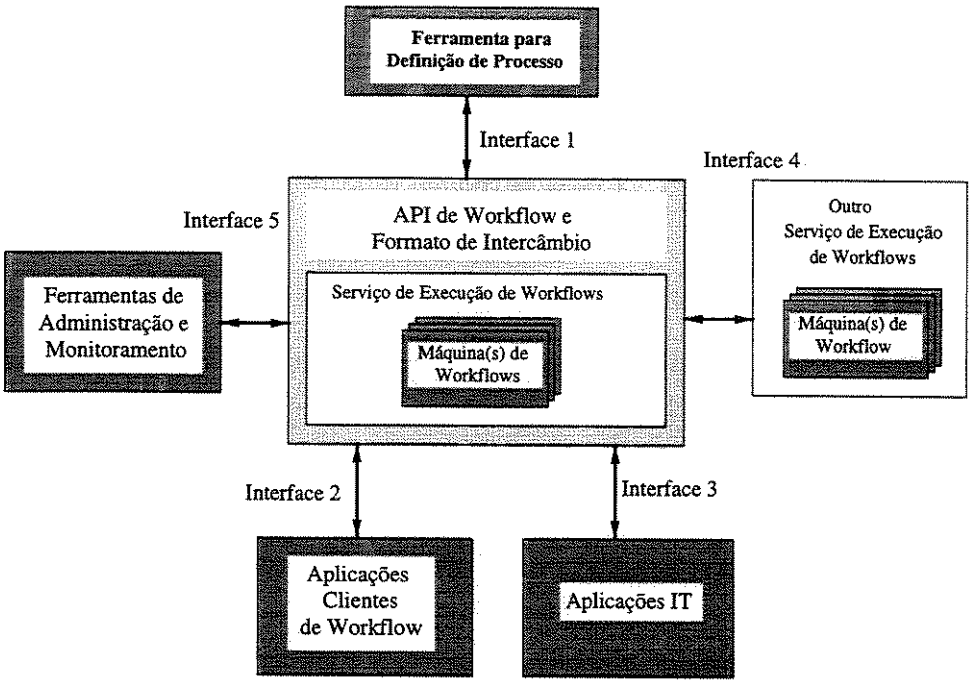

Figura 3.3 - Componentes e interfaces [6]

A WfMC, com seus membros, não é a única a querer definir padrões para os sistemas de gerenciamento de workflow. Outras empresas que possuíam produtos diferentes do modelo proposto pela WfMC não aceitaram a idéia de ter que adaptar o seu produto ao padrão proposto, o

que levou a um impasse do problema que impossibilitou a resolução do problema naquele momento.

No início de 1997, a OMG (Object Management Group) publicou um documento denominado "Workflow Facility Request For Proposal" [10], para tentar resolver o problema acima. Este documento era um apelo para os fabricantes de produtos workflow para submeterem suas propostas à análise tendo em vista a criação de um modelo capaz de definir interfaces e suas respectivas semânticas, visando a interoperabilidade de objetos de workflow e seus metadados.

Este modelo é denominado de *Sistema para Gerenciamento de Workflows* (Workflow Management Facility) e serve como uma plataforma de integração para a construção de sistemas de gerenciamento de workflow com interfaces bem definidas e para incorporar outras aplicações e objetos legados.

Quatro propostas foram submetidas em primeira instância: (i) Nortel [13]; (ii) jFlow [12]; (iii) EDS [11]; e ISTI. As propostas foram avaliadas por um comitê, críticas foram feitas e os grupos replicaram. Apenas dois grupos permaneceram após esta primeira fase: (i) a Nortel, que apresentou um documento muito robusto e completo; (ii) e o grupo jFlow, que acabou incorporando a EDS e lançou uma proposta consideravelmente diferente da primeira.

3.2. jFlow

A submissão da jFlow está apoiada por mais de 20 empresas e por outros 9 membros da OMG, sendo que um deles é a WfMC (Workflow Management Coalition). Por não ser uma companhia com um produto próprio, não foi permitido à WfMC submeter seu modelo de referência. Desta maneira, os seus membros o fizeram. A maioria dos participantes tem um produto de gerenciamento de workflow ou faz uso em larga escala de um produto que em maior ou menor grau é compatível com modelo de referência da WfMC [6].

A proposta da jFlow aborda principalmente os aspectos referentes à interação do gerenciador de workflow com os recursos, manuais ou automáticos, existentes no ambiente de trabalho, interação entre WFMSs e monitoramento e auditoria de uma determinada instância de workflow. No entanto, os problemas relacionados à manipulação e gerenciamento de definições de processo (esquemas workflow) foram deixados para futuras versões. No contexto desta proposta, um processo de workflow é o mesmo que instância de workflow, como discutido em seções anteriores.

A figura 3.4 mostra o diagrama de interface proposto pela jFlow seguido de um pequeno comentário sobre os componentes mais importantes. Informações detalhadas podem ser obtidas diretamente do documento [12]. Apesar de ser nítida a distinção entre classe e interface, utilizam-se os termos indistintamente, pois, da maneira como são referidas o que importa são as semelhanças dos conceitos que suportam (relacionamentos, herança, cardinalidade).

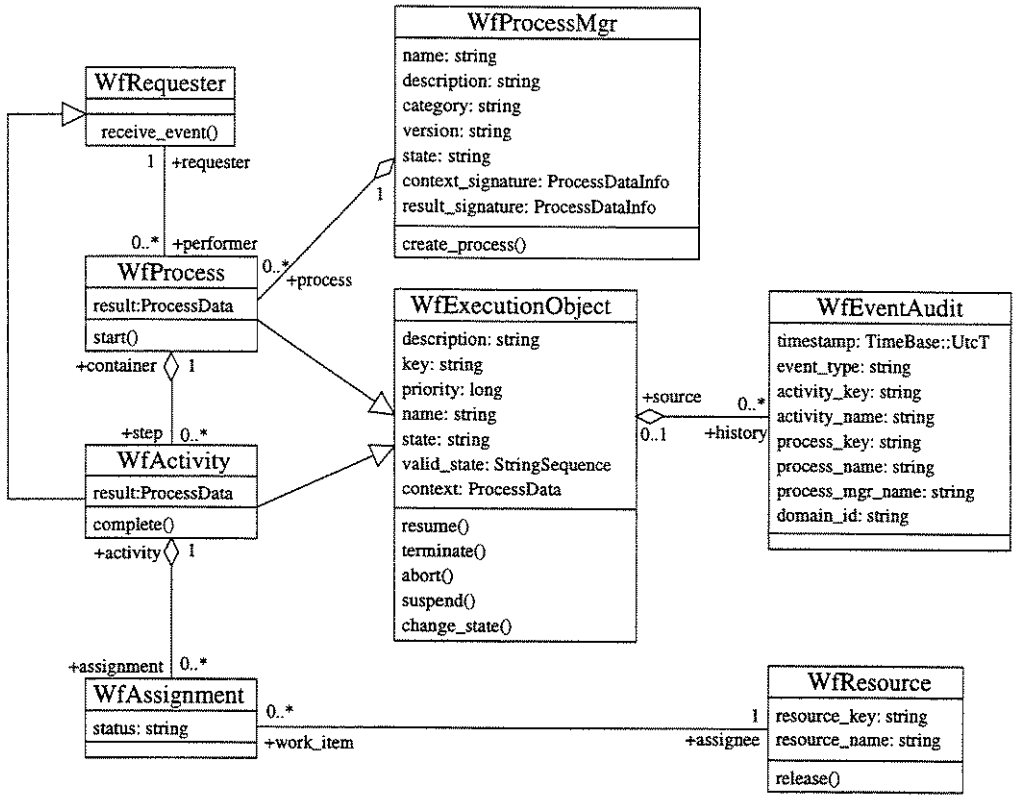

Figura 3.4 - Modelo de interfaces da jFlow [12]

O *Requisitante de Workflows* (WfRequester) é a interface que se preocupa diretamente com a execução e os resultados de um workflow - representa a requisição de algum trabalho que deve ser feito. Ele espera de quem irá realizar este trabalho que manipule a requisição e comunique as mudanças de estado significantes, e.g., informar o requisitante quando o trabalho estiver completado, etc. Um requisitante pode ter muitos processos associados a ele.

Normalmente, o *Requisitante de Workflow* é quem inicia *Processo de Workflow* (WfProcess). Durante esta fase algumas ações de controle devem ser feitas, tais como, definir os dados de contexto (context data) para iniciar o processo e recuperar os resultados ou estados durante o seu ciclo de vida.

Há dois cenários da associação entre um *Processo de Workflow* e um *Requisitante de Workflow*:

(i) permitir que um *Requisitante de Workflow* delegue a execução de um processo pelo *Processo de Workflow*.

(ii) aninhamento de processos: utilizando-se as operações herdadas da interface *Requisitante de Workflow*, uma *Atividade de Workflow* (WfActivity) requisita a um *Processo de Workflow* a

execução de um determinado trabalho. No final da execução deste, a *Atividade de Workflow* recebe as informações sobre o estado do processo requisitado. Isto permite a construção recursiva de estruturas de workflow, como exemplificado na figura 3.5.

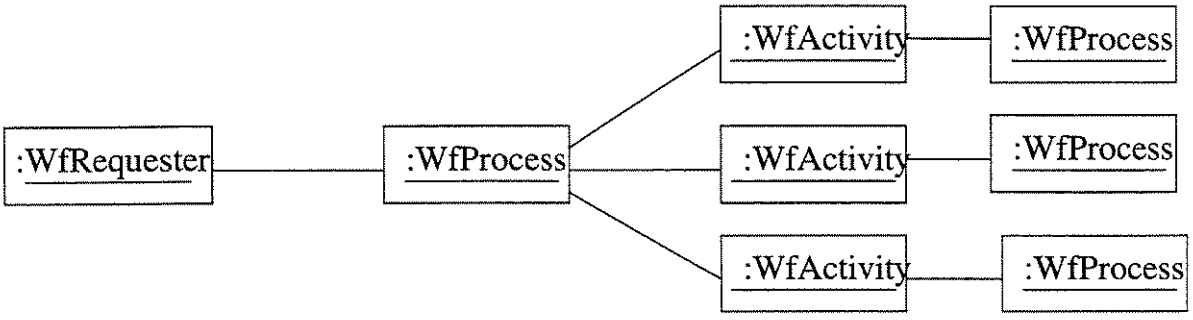

Figura 3.5 - Exemplo de um workflow aninhado

O *Objeto de Execução de Workflow* (WfExecutionObject) é uma interface base abstrata que define atributos, estados e operações comuns para o *Processo de Workflow* e *Atividade de Workflow*. São definidas operações para: (i) adquirir e definir o seu estado interno, i.e., os valores dos seus atributos; (ii) fazer transições específicas tais como suspender-se (*suspend*), continuar executando a partir do ponto onde foi suspendida (*resume*), terminar a execução imediatamente (*terminate*), e abortar a execução (*abort*); (iii) monitorar o WfExecutionObject retornando, baseado em filtros, registros de eventos que representam o histórico de sua execução; (iv) definir e adquirir o contexto. Também define atributos de nome (*name*), descrição (*description*) e identificador unívoco do objeto (*key*). A figura 3.6 mostra os principais estados que um objeto de execução pode assumir durante seu ciclo de vida.

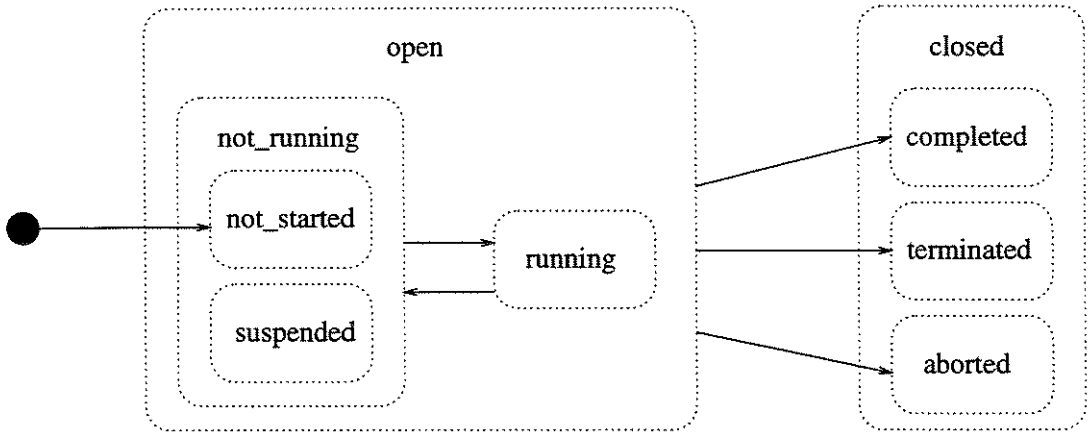

Figura 3.6 - Estados de um objeto de execução [12]

O *Gerenciador de Processo de Workflow* (WfProcesMgr) define a interface para dar suporte a fabricação e localização de *Processos de Workflow*. Também provê operações para o acesso de

meta-informações produzidas ou exigidas por um *Processo de Workflow*. O nome do *Gerenciador de Processo* deve ser único dentro do domínio onde estiver sendo utilizado. Desta maneira, ele pode ser localizado usando-se o OMG Naming Service [1] ou através do nome e outros atributos (e.g., categoria) através do OMG Trader Service [1] ou outros mecanismos de infra-estrutura.

O *Processo de Workflow* (WfProcess) é uma interface que todo objeto que realize algum tipo de trabalho deve implementar. Permite ao trabalho proceder assincronamente enquanto estiver sendo monitorado e/ou controlado. Ele pode conter zero ou mais passos (relacionamento *step*) que são representados pelas *Atividades de Workflow*. Para que um processo de negócio seja automatizado deve haver um *Processo de Workflow* que implemente sua lógica, ou um genérico que sirva como um máquina de workflows. Ele deve ser alimentado, durante sua iniciação, com a definição de processo e outros dados de contexto. Adicionalmente, são definidas as seguintes operações para (i) iniciar (*start*) a execução, (ii) obter os resultados produzidos, (iii) recuperar informações do relacionamento entre ele e um *Requisitante de Workflow* ou *Atividade de Workflow*, (iv) monitorar o estado ou fazer o controle do processo, são definidas.

A *Atividade de Workflow* (WfActivity) é um passo dentro de um *Processo de Workflow*. Na verdade esta interface é utilizada para se construir workflows compostos. Basicamente ela é utilizada para requisitar trabalho de outro *Processo de Workflow* e *Recurso de Workflow*. Pode haver muitas *Atividades de Workflow* ativas dentro de um determinado *Processo de Workflow* em um determinado instante. Na proposta não se especifica o momento de criação delas atividades, podendo estas ser criadas durante a instanciação do processo que a contém ou sob demanda.

O *Recurso de Workflow* (WfResource) é uma interface para representar um pessoa ou coisa que realize o trabalho de uma Atividade de Workflow forma semi-automática.

A *Associação de Workflow* (WfAssignment) liga uma Atividade de Workflow ao *Recurso de Workflow*.

3.2.1. Cenário de Execução de um Processo

Para iniciar a execução de um determinado *Processo de Workflow* é necessário identificar-se o *Requisitante de Workflow* responsável por ele. Um *Requisitante* pode ser criado especificamente para o *Processo* em questão ou um já existente pode ser reutilizado.

A criação e associação do *Processo de Workflow* ao *Requistante de Workflow* é feita pelo *Gerenciador de Processo de Workflow* (WfProcessMgr). As implementações desta interface permitirão localizar e instanciar *Processos de Workflow*. Não é responsabilidade desta entidade gerenciar a execução de uma determinada instância de workflow. A entidade que dá suporte a esta funcionalidade tem que implementar a interface *Processo de Workflow*. Fazendo analogia com o modelo da WfMC, o *Gerenciador de Processo de Workflow* é um Sistema de *Execução de*

Workflows (Workflow Enactment Service) e a máquina de workflows deve implementar a interface *Processo de Workflow*.

Os processos precisam de um conjunto de dados mínimo para iniciar sua execução. Denomina-se dados de contexto ou apenas contexto as informações que são processadas durante a execução do *Processo de Workflow*, as informações sobre a definição de processo que será utilizado.

Uma vez satisfeita as premissas anteriores, o processo pode ser disparado (*started*) através da operação *start*. A partir deste ponto, todo o controle e a coordenação das tarefas são feitos pelo processo que implementa o serviço de execução.

Quando uma *Atividade de Workflow* for ativada, seus dados de contexto são definidos e seus recursos podem ser associados por meio da criação de *Associações de Workflow* entre ela e os *Recursos de Workflow* que irão utilizar. O mecanismo de seleção de recursos não foi definido na proposta, porém uma aplicação pode usar um outro processo, gerenciador de recursos, que tenha a função de encontrar o recurso disponível no ambiente com base nas informações de contexto da atividade e outros parâmetros do processo.

Durante a execução as *Atividades de Workflow* são ativadas com base nos dados produzidos por elas e na lógica do *Processo de Workflow*. A interface *Atividade de Workflow* na verdade serve como um requisitante de trabalho e apenas invoca outros processos como seus sub-processos.

Um *Processo de Workflow* estará completo quando não houver mais nenhuma *Atividade de Workflow* a ser ativada. Neste ponto, o *Processo de Workflow* informa o *Requisitante de Workflow* do seu estado corrente. Uma vez informado o *Requisitante de Workflow* pode recuperar os dados produzidos.

O *Requisitante de Workflow*, além de ser notificado das mudanças significativas de estado do *Processo de Workflow* requisitado pode obter informações sobre o estado global do processo através das operações *state*, *get_context* e *get_result*. Informações mais detalhadas do estado dos diversos passos do processo podem ser obtidas navegando-se pelo relacionamento *step* entre um processo e suas atividades, e utilizando-se os resultados das consultas providas pela interface WfActivity. A navegação em workflows aninhados é apoiada pelo relacionamento *process* entre uma WfActivity (no papel de requisitante) e os seus sub-processos.

Toda vez que um Objeto de Execução executar uma mudança de estado, relevante ao workflow, um *Evento para Auditoria de Workflow* (WfEventAudit) é registrado. Para cada *Objeto de Execução de Workflow*, pode-se acessar os eventos, por ele gerados, para analisar o histórico de sua execução. Os eventos podem ser publicados usando-se o OMG Notification Service [1].

3.3. Nortel

A proposta da Nortel foi patrocinada pela Northen Telecom e construída pela University of Newcastel em Tyne na Inglaterra. Existe um protótipo funcional que implementa parte proposta [27][25].

A proposta da Nortel foi estruturada independentemente de qual seja o ambiente onde ela possa ser empregada, suportando a automação de processos de negócio das mais variadas naturezas. Isto foi possível porque foi construída com base num meta-modelo de processo de alto nível que não favorece nenhuma modelagem ou metodologia específica de definição de processo. Este meta-modelo permite que a proposta seja estendida para suportar as necessidades inerentes a qualquer processo de negócio.

Na terminologia adotada, uma aplicação de workflow (ou instância de workflow) é uma coleção de tarefas interdependentes. Uma tarefa é uma unidade elementar de trabalho que possui relevância no ambiente onde o sistema de gerenciamento de workflows for empregado.

A proposta está divida em dois sub-sistemas: (i) sistema de execução (*execution facility*), que suporta a execução de workflows; (ii) sistema de repositório (*repository facility*), que suporta o armazenamento e recuperação de definições de processo.

O *modelo de tarefa* é um mecanismo interessante para a execução de workflows, tanto para a criação de definições de processo quanto para a criação do mecanismo de execução da máquina de workflows (workflow engine).

No *modelo de tarefa*, uma tarefa (*task*) é composta de um ou mais conjuntos de entradas (*input set*) e um ou mais conjuntos de saídas (*output set*). Como mostrado na figura 3.7, a tarefa t_1 tem 3 conjuntos de entrada (I_1, I_2 e I_3), e dois conjuntos de saída (O_1 e O_2). Cada conjunto de entrada possui um certo número de recursos de entrada (*input resources*) que são consumidos durante a execução da tarefa. Uma vez que todos os recursos de entrada de um determinado conjunto de entrada estiverem disponíveis (incluindo entradas virtuais que representam notificações temporais) a tarefa está apta a iniciar a execução. Se ocorrerem casos em que múltiplos conjuntos de entrada tornarem-se disponíveis ao mesmo tempo, então o conjunto de maior prioridade deve ser escolhido para a execução. Os conjuntos de saída (*output sets*) são conjuntos mutuamente exclusivos de recursos de saída (*output resource*), que as tarefas produzem após o seu término. Toda vez que uma tarefa terminar sua execução, um e apenas um conjunto de saída estará disponível para uso. A partir deste momento, as tarefas que são dependentes do conjunto de saída produzido são notificadas desta disponibilidade, desenrolando desta maneira a execução do workflows [13].

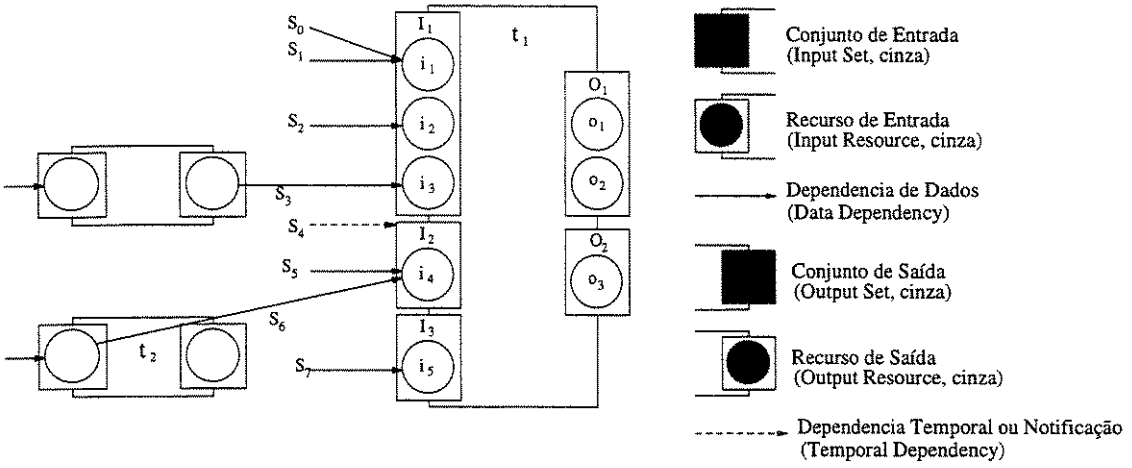

Figura 3.7 - Notação de uma tarefa do modelo da Nortel [13]

As dependências entre as tarefas podem ser de dois tipos: (i) dependências de dados, são aquelas onde a ativação de uma determinada tarefa ocorre quando os dados que ela estiver esperando estiverem disponíveis (foi produzido ou foi atribuído como um insumo); (ii) notificações, também denominadas de dependências temporais, são utilizados para estabelecer a precedência temporal entre as tarefas. Propõe-se que as notificações sejam implementadas como recursos que não possuam valor algum.

Denominam-se alternativas de entrada (*input alternative*) às fontes alternativas de recursos que satisfazem um determinado recurso de entrada. O recurso (figura 3.8), que é a fonte de uma alternativa de entrada, origina-se, normalmente, de um conjunto de saída, embora também possa ser originado de um conjunto de entrada.

A proposta foi projetada para suportar quatro tipos de tarefas, que juntas permitirão a especificação e execução de processos de negócio complexos:

(i) tarefa simples (*simple task*), que representa uma unidade elementar de trabalho dentro do workflow (figura 3.9-a);

(ii) tarefa composta (*compound task*), que permite a composição da tarefa com outras tarefas. Desta maneira podem ser criadas estruturas recursivas de aplicações workflows. Um processo de negócio será representado por uma tarefa composta. A figura 3.9-b mostra uma tarefa composta t_1 contituídos pelas tarefas t_2 e t_3;

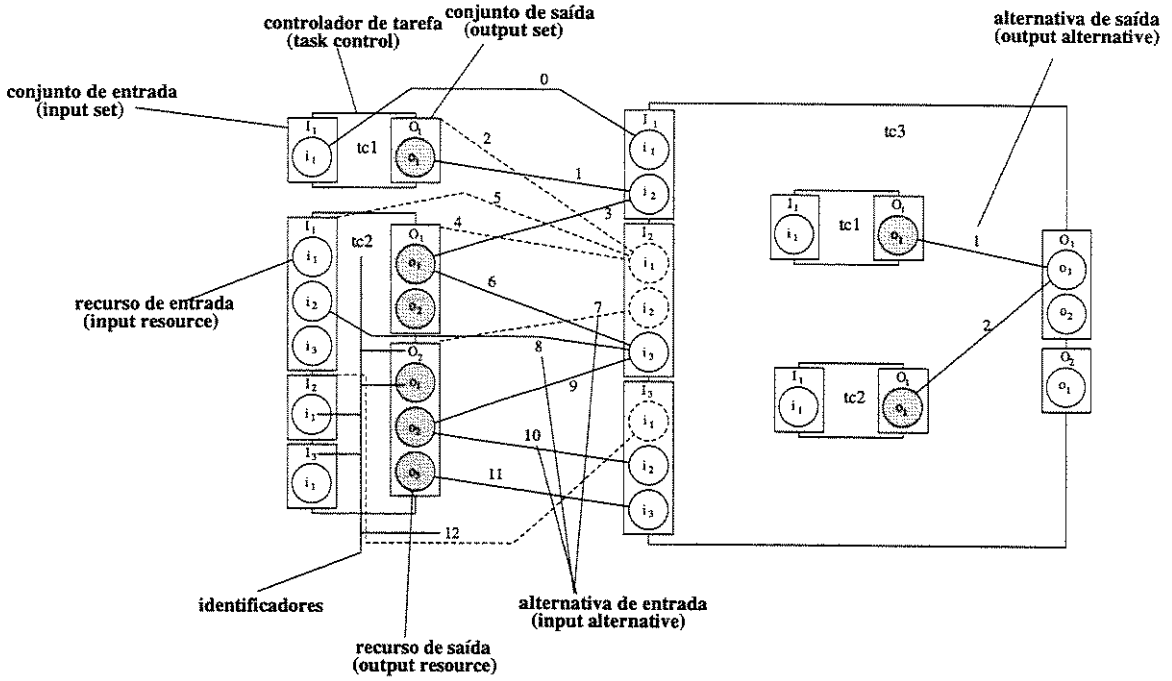

Figura 3.8 - Componentes do modelo de tarefa da Nortel [13]

(iii) tarefa genesis (*genesis task*), que tem como principal finalidade a instanciação dinâmica (sob demanda) de tarefas. Elas são utilizadas para se conseguir o seguinte: (iii.i) aplicações workflows que contenham execuções recursivas, isto é, uma estrutura de tarefa cuja execução causa a execução recursiva de sua própria estrutura, em tempo de execução do workflow, como sua sub-tarefa; (iii.ii) instanciação dinâmica de partes de uma aplicação workflow (um caso degenerado do acima). A estruturação de uma aplicação em termo de "genesis task" provê um meio eficiente de se gerenciar workflows muito grandes, pois somente as partes estritamente necessárias são instanciadas. (figura 3.9-c);

(iv) tarefa adaptadora (adapter task), será utilizada para a construção de *gateways* para sistemas de gerenciamento de workflow legados. (figura 3.9-d).

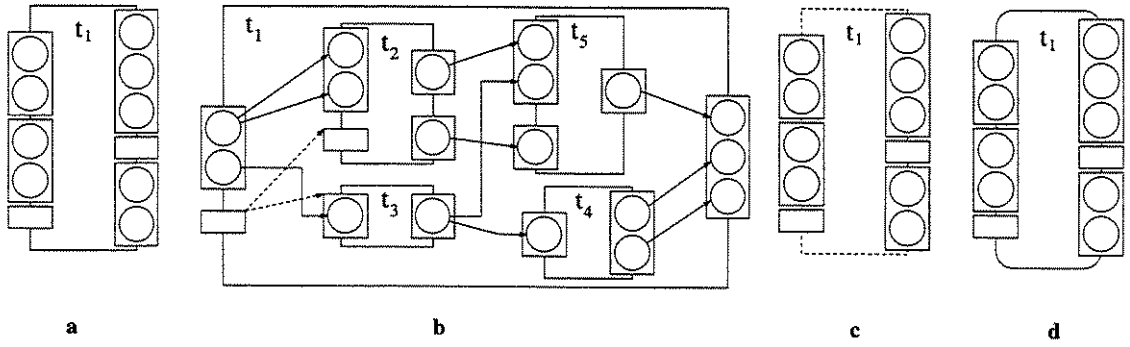

Figura 3.9 - Notação para os tipos de tarefas da Nortel [13]

O sistema de execução de workflows (*workflow execution facility*) registra as dependências entre as tarefas em objetos persistentes e requer que as transações para propagar informações de coordenação sejam atômicas. O modelo de transação proposto nesta especificação é pobre, pois ele somente é adequado a processos de curta duração e reversíveis.

O sistema de execução é composto de controladores de tarefa (*task controllers*) e tarefas (*tasks*). A principal função do controlador de tarefas é disparar a execução do(s) objeto(s), associado(s) a ele, que implementa(m) a interface tarefa. Além disso provê operações para gravar e manipular informações sobre as alternativas de entrada (*input alternatives*), alternativas de saída (*output altenatives*), tarefas associadas (*tasks*), fábrica de tarefas (*task factory*) e fábrica de controladores de tarefas (*task control factory*). Também provê operações para a requisição de notificações (*request notification*) e notificação (*notification*). A operação de requisição de notificação normalmente é invocada, durante o *setup* da aplicação, em uma instância do controlador de tarefas por outra instância da mesmo tipo para registrar a dependência (tanto dependência de dados quanto temporal) entre as duas. Esta associação também pode ser feita durante a execução da aplicação, como é o caso das tarefas genesis, permitindo dinamizar a estrutura das dependências. A figura 3.10 mostra o diagrama de interfaces proposto pela Nortel para o sistema de gerenciamento de workflow.

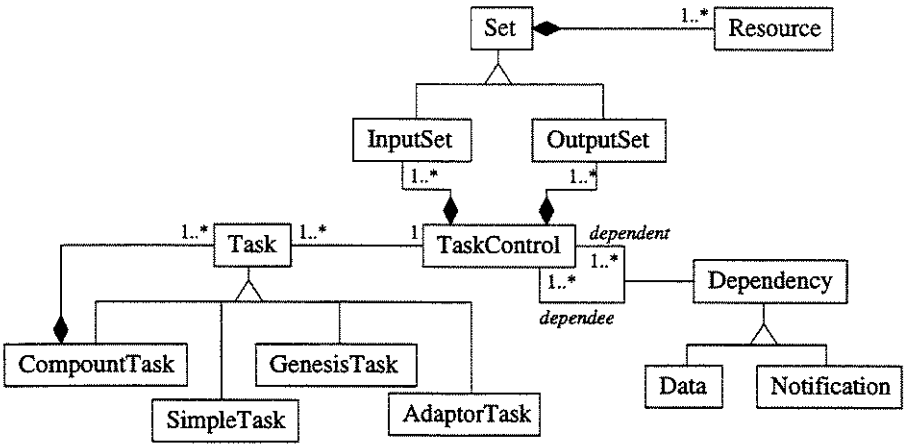

Figura 3.10 - Diagrama de interfaces da Nortel

A interface tarefa é muito parecida com a interface processo (WfProcess) da proposta da jFlow. A operação mais importante desta interface é a operação *start*, que recebe alguns parâmetros como a referência do controlador de tarefas a ela, informações de entrada na forma de seqüência de referências a recursos (*resources*) e identificadores de conjuntos de entrada (*input set*).

Cada tarefa dentro de uma aplicação de workflow possui um controlador de tarefa associado a ela. Sua finalidade é receber notificações de saída, de outros controladores, e usá-las para determinar quando a tarefa, associada a ele, pode ser disparada (*started*). Ele também é responsável

pela propagação de notificações das saídas produzidas, por suas tarefas, para os outros controladores interessados. O controlador armazena também informações sobre o estado (*status*) da tarefa, importantes para o seu monitoramento.

A figura 3.11 mostra um possível cenário da distribuição dos componentes de um WFMS dentro do modelo da Nortel.

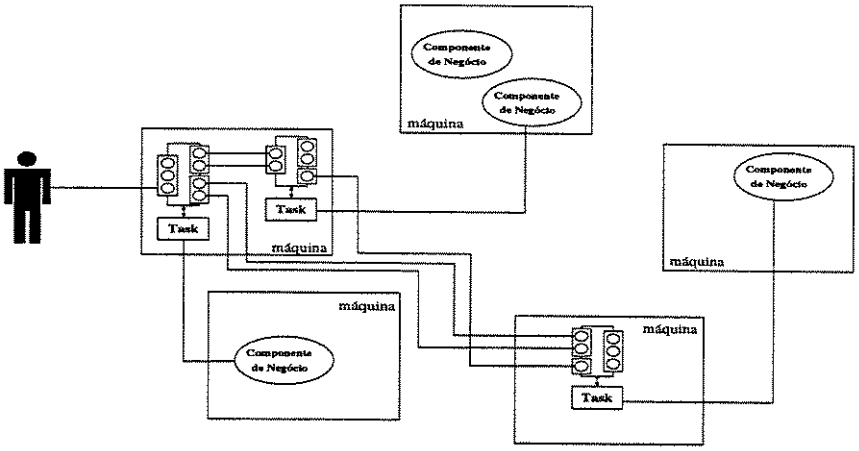

Figura 3.11 - Possível distribuição dos componentes de um sistema de gerenciamento de workflows

A associação entre controladores de tarefa e tarefas pode ser feita de diversas maneiras:

(i) coordenação distribuída é aquela em que um controlador de tarefa está localizado no mesmo lugar (e.g., máquina de workflows) onde está a tarefa a ele associada (figura 3.12-a);

(ii) coordenação centralizada é aquela na qual todos controladores de tarefa são agrupados em uma máquina de workflows (figura 3.12-c);

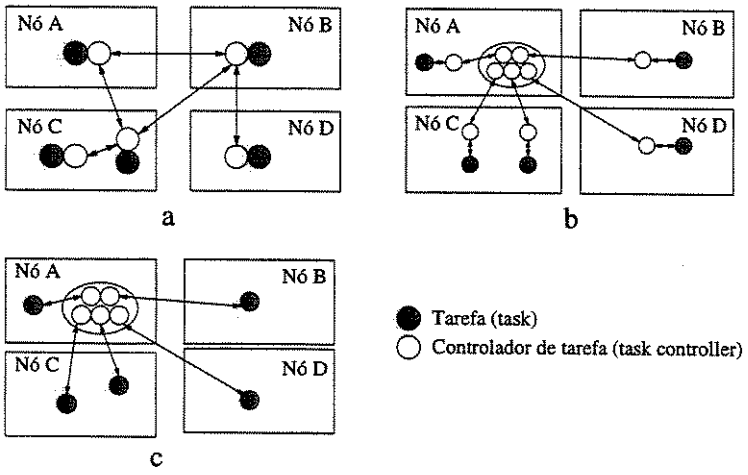

Figura 3.12 - Distribuição de tarefas e controladores de tarefas [13]

(iii) coordenação híbrida é aquela onde todos os controladores de tarefa estão agrupados em uma determinada máquina de workflows mas coordenam outros controladores que estão juntos de suas tarefas em outra localidade (este esquema pode ser feito usando-se compound tasks), figura 3.12-b.

A escolha da distribuição adequada deve ser feita pelo responsável pela aplicação de workflows ou por um administrador. Diversos fatores podem influenciar tal seleção. Entre eles destacam-se o grau de tolerância a falhas exigido, conveniências administrativas, etc. Propõe-se que a notificação entre os controladores de tarefa seja realizada como uma transação. As tarefas podem também ser executadas numa transação.

A facilidade de reposição (*repository facility*) armazena toda informação relevante de um esquema de workflow esquema: informações das tarefas simples, compostas, genesis e adaptadoras, conjuntos de entrada e saída, alternativas de entrada e saída. O repositório provê um conjunto de operações transacionais para manipulá-los.

Capítulo 4

Protótipo do Sistema de Gerenciamento de Workflows

Este capítulo mostra o projeto de um protótipo coerente e experimental de um sistema de gerenciamento de workflows que combina versões simplificadas do modelo proposto pela jFlow [12], do modelo de tarefa proposto na especificação da Nortel [13] e do modelo de um sistema de gerenciamento de workflows genérico apresentado no modelo de referência da WfMC [6].

O protótipo foi construído para:

(i) servir como infra-estrutura para experimentos futuros;

(ii) verificar a viabilidade de construir um sistema de gerenciamento de workflows compatível com a proposta da jFlow;

(iii) verificar a complexidade da construção de uma *Máquina de Workflows* e linguagem de definição de processo orientados a eventos.

Para fazer uma apresentação didática, o protótipo está apresentado em diferentes níveis de detalhes. No primeiro nível, encontra-se um modelo mostrando as principais entidades que compõem o sistema de gerenciamento de workflows proposto e uma descrição de suas responsabilidades. No segundo nível, encontra-se um modelo mostrando um possível mapeamento das entidades do modelo do primeiro nível nas interfaces especificadas na proposta da jFlow e o principal cenário da utilização do protótipo, qual seja, a execução de processos. No terceiro nível, encontra-se a especificação dos atributos e comportamentos das operações, dando ênfase às entidades consideradas principais dentro do sistema de gerenciamento de workflows. Todos os modelos criados foram especificados segundo a notação UML [17][22] e para um melhor entendimento aconselha-se fazer a referência ao Apêndice C, pois há um breve resumo desta notação.

4.1. Sistema de Gerenciamento de Workflows Genérico (Nível 1)

A figura 4.1 mostra o diagrama de entidades de primeiro nível. Ele foi obtido com base na figura 3.2, que mostra as entidades geralmente contidas em um sistema de gerenciamento de workflows. O modelo de um sistema de gerenciamento workflows genérico, figura 3.2, foi escolhido como a base do modelo da figura 4.1 por ser um documento público e conter uma descrição completa e didática dos componentes que compõem um sistema de gerenciamento de workflows.

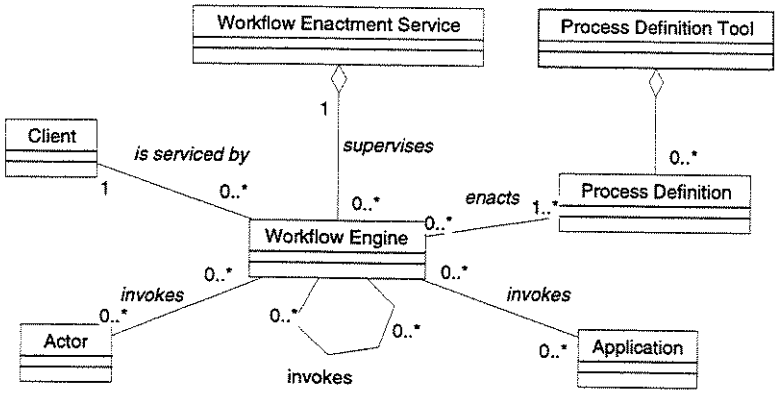

Figura 4.1 - Modelo de um sistema de gerenciamento de workflows genérico

Um *Sistema de Gerenciamento de Workflows* (Workflow Management System) é um termo utilizado para designar o conjunto de ferramentas e conceitos que permitem a definição, criação, gerenciamento, supervisão e monitoramento da execução automática de workflows. Seus principais componentes são: (i) *Serviço de Execução de Workflows* (Workflow Enactment Service), responsável por fornecer serviços para o gerenciamento, supervisão e monitoramento da execução automática de workflows; (ii) *Ferramenta para a Definição de Processos* (Process Definition Tool), responsável por fornecer serviços para a definição, criação e simulação de definições processos.

O *Cliente* (Client) é a entidade que possui a necessidade da execução de um determinado processo. Inicia (fornece as informações necessárias para ativar a execução do processo) e possivelmente monitora a transação e a ele devem ser encaminhados os resultado produzidos.

O *Serviço de Execução de Workflows* (Workflow Enactment Service) é a entidade que fornece os serviços de execução e localização de workflows para os *Clientes*. O *Serviço de Execução de Workflows* pode, em um determinado instante, estar executando workflows para diversos *Clientes*. Cada workflow em execução é chamado de instância de workflow e é manipulado por uma *Máquina de Workflows* (*Máquina de Workflows*) distinta.

A *Máquina de Workflows* é a entidade que controla a execução de uma instância de workflow (processo) para um *Cliente*. A coordenação e o controle da execução dos processos são feitos pela interpretação da *Definição de Processo* (*Definição de Processo*) e delegação da execução das tarefas que compõem o processo às entidades executoras (*Ator*, *Aplicação* ou outra *Máquina de Workflows*) competentes para tal.

O *Ator* (Actor) é a entidade que representa uma pessoa ou grupo de pessoas que executam uma dada tarefa, pertencente ao processo, de forma semi-automática (pessoas ou grupo de pessoas interagindo com programas de aplicação ou recursos IT). Uma caraterística importante desta entidade é sua disponibilidade, devendo haver, por isso, (embora não implementado neste protótipo) algum mecanismo de verificação da disponibilidade para o trabalho destas entidades. A cada *Ator* está associado um conjunto de papeis (roles) que ele pode desempenhar. Esta informação é utilizada para fazer a atribuição de tarefas aos *Atores* existentes.

Uma *Aplicação* (*Aplicação*) é a entidade que representa qualquer coisa que execute uma determinada tarefa, pertencente ao processo, automaticamente. Numa implementação ela deve ser representada pelos recursos e aplicações IT existentes no ambiente onde o processo existe. A elas também estão associados os papeis por elas desempenhados, da mesma forma como foram associados aos *Atores*. A preocupação com a disponibilidade de uma *Aplicação* não é tão relevante quanto a de um *Ator*.

A *Ferramenta para a Definição de Processo* (*Definição de Processo* Tool) é a entidade responsável em prover facilidades para a criação de *Definições de Processo*. Normalmente, as ferramentas são desenvolvidas por projetistas de programas para a reengenharia de processos. Elas também fornecem facilidades para a visualização de simulações, testes, etc, de processos. Embora esta entidade seja de grande importância dentro do contexto do gerenciamento de workflows, ela não foi implementada no protótipo.

A *Definição de Processo* (*Definição de Processo*) é a entidade que possui a lógica de um determinado processo. Nela estão especificadas quais são as tarefas que constituem o processo, as dependências entre elas, as condições para ativá-las e o papel do recurso (*Ator*, *Aplicação* ou outra *Máquina de Workflows*) que realizará o trabalho representado. A *Definição de Processo* define o comportamento da *Máquina de Workflows* no momento de execução do workflow: desta maneira, basta alterá-la para que a *Máquina de Workflows* se comporte de forma diferente. A *Definição de Processo* é criada através de uma linguagem de definição de processos que provê o conjunto de conceitos necessários para se modelar o processo.

A separação das informações de controle (*Definição de Processo*) das entidades que realizam as tarefas (*Ator, Aplicação* e *Máquina de Workflows*) conduz à facilidade de criação e alteração dos processos, pois o controle é considerado a característica do processo que tende a mudar com freqüência, enquanto que as entidades que realizam o trabalho das tarefas são menos susceptíveis a alterações [21].

A descrição anterior mostra as entidades que compõem o protótipo do sistema de gerenciamento de workflow de uma forma estática. Nada foi dito sobre a interação destas entidades antes, durante e após a execução de um processo. Para evitar redundâncias, a apresentação do modelo que mostra a dinâmica entre estas entidades foi adiada para o segundo nível de detalhamento. Isto permite a descrição do cenário em termos das operações expostas por aquelas entidades após terem sido *supertipadas* pelas interfaces propostas no modela da jFlow.

4.2. Modelo Combinado de um WFMS (Nível 2)

A figura 4.2, que pertence ao segundo nível de detalhamento, mostra o modelo obtido pela fusão do modelo da figura 4.1 com uma versão simplificada do modelo da jFlow.

O principal motivo pelo qual o modelo da jFlow foi escolhido para ser mesclado com o modelo do primeiro nível de detalhes, foi o fato de ter sido adotado pela OMG para ser o modelo de Workflow Management Facility de sua arquitetura.

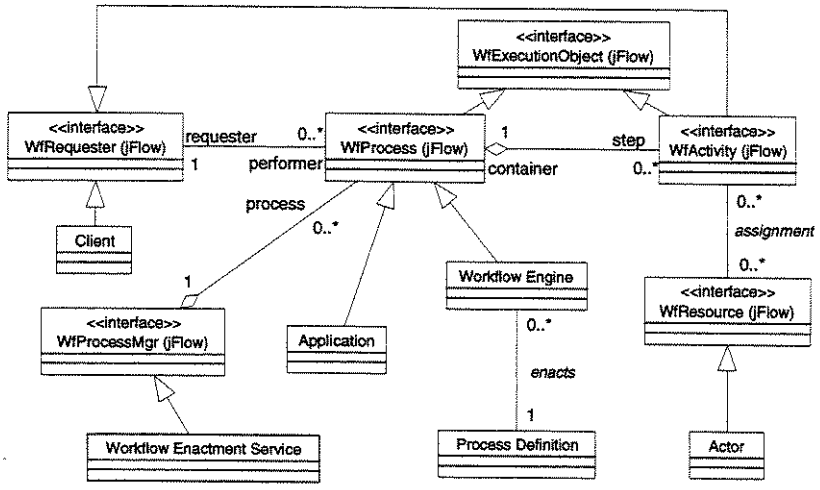

Figura 4.2 - Modelo combinado

Segundo a proposta da jFlow, toda entidade que realize qualquer trabalho (tarefa) automaticamente deve herdar a interface *WfProcess*. Entre as operações providas por esta interface estão aquelas que permitem definir os dados de iniciação, iniciar a execução de

WfProcesss, além de permitir a visibilidade do *WfRequester* associado a ele. Por estas características, as entidades *Máquina de Workflow* e *Aplicação* herdaram essa interface.

A interface *WfProcessMgr* provê operações para criar e localizar a instância do workflow. Desta maneira a entidade *Serviço de Execução de Workflows* herda e implementa esta interface, possibilitando a ela criar e localizar *Máquinas de Workflow*.

A entidade *Ator* herda a interface *WfResource* porque ela foi concebida de tal maneira a permitir às pessoas ou grupo de pessoas participarem na execução de processos.

A entidade *Cliente* herda a interface *WfRequester* porque essa fornece as operações e relacionamentos necessários para que ele inicie uma instância de *WfProcess*, consulte e seja notificado sobre o seu estado durante sua execução.

Pela *supertipagem* (*supertyping*) das entidades do modelo do primeiro nível com as interfaces da jFlow pode-se criar o cenário mais importante de um sistema de gerenciamento de workflows, i.e., a execução de processos, utilizando-se as operações herdadas.

A figura 4.3 apresenta um possível cenário da interação das entidades que compõem o sistema de gerenciamento de workflows, utilizando-se o diagrama de colaboração UML [17][22], durante o ciclo de execução de um workflow. Logo abaixo encontra-se uma descrição de acompanhamento para melhor compreender os passos realizados.

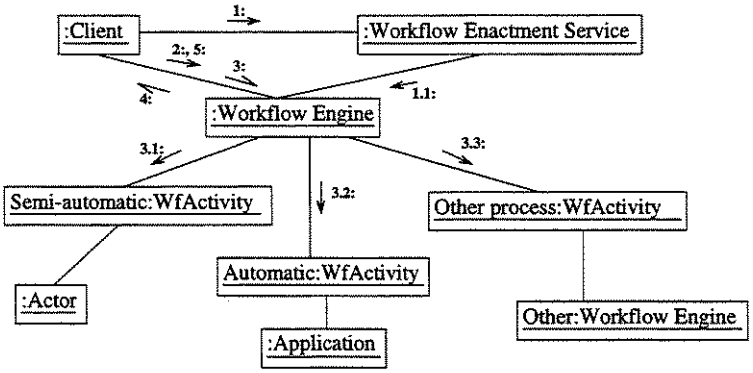

Figura 4.3 - Diagrama de colaboração da execução de um processo genérico

1: [`WfProcess create_process(WfRequester requester)`]. O *Cliente* utiliza esta operação para iniciar uma instância de workflow. De alguma maneira, e.g., através do ORB (Object Request Broker) ou servidor de nomes, o *Cliente* deve adquirir a referência do *Serviço de Execução de Workflows* para invocar esta operação. O parâmetro passado é a referência do requisitante (`requester`) do processo, que normalmente é a referência do próprio *Cliente*. Isto permite à *Máquina de Workflows* comunicar-se com o

seu requisitante. O retorno desta operação será a referência à *Máquina de Workflows* que foi criada, permitindo ao *Cliente* realizar a consulta e alteração do estado da *Máquina de Workflow*.

1.1: [`new( WfRequester requester)`]. Esta operação cria uma instância de um *Máquina de Workflow*. A referência, passada como parâmetro, é utilizada para definir a associação entre a *Máquina de Workflows* e o *Cliente*, permitindo, ao primeiro, comunicar as suas mudanças de estado ao segundo.

2: [`void set_context(ProcessData new_context)`]. Através desta operação, o *Cliente* define os *dados de contexto* da *Máquina de Workflows*. Os *dados de contexto* são compostos do nome da *Definição de Processo* que será executada e o conjunto de dados iniciais que são processados durante a execução do processo.

3: [`void start()`]. Esta operação dispara o *Máquina de Workflow* que ativa as tarefas que devem ser desempenhadas de acordo com a *Definição de Processo*. É uma operação assíncrona, i.e., o controle de execução retorna imediatamente ao *Cliente* após invocar a operação, permitindo a execução paralela da *Máquina de Workflow* e do *Cliente*.

3.1 (3.2, 3.3): [`new(); set_context(ProcessData value); change_state()`]. A *Máquina de Máquina de Workflows* cria instâncias de *WfActivitys* (`new()`), define seus *dados de contexto* (`set_context(ProcessData value)`) e ativa suas respectivas execuções alterando-se seus estados (`change_state()`). A realização de cada *atividade* (representada pelas *WfActivitys*) é delegada a um *Ator*, *Aplicação* ou outra *Máquina de Workflows* de acordo com o *papel* (*role*) especificado na *Definição de Processo*.

4: [`void receive_event(ProcessData event)`]. Esta operação serve para informar ao *Cliente* do término do processo por ele requisitado.

5: [`ProcessData result()`]. Com esta operação o *Cliente* recupera os dados produzidos pela execução do processo.

O cenário acima apresenta uma visão da interação das entidades durante a execução de um processo de um ponto de vista bastante abstrato. Entretanto nada foi dito sobre o mecanismo que controla e coordena a ativação das atividades que compõem um processo. Estas informações estão esclarecidas no próximo nível de detalhamento.

4.3. Especificação do Modelo Combinado (Nível 3)

O terceiro nível de detalhamento é caracterizado por apresentar informações muito próximas ao que deve ser a implementação de algumas das entidades acima citadas.

Das interfaces herdadas do modelo da jFlow apenas a *WfActivity* recebeu uma implementação. As outras foram utilizadas principalmente para herdar-se o conjunto de operações definidas.

O *Máquina de Workflows* e a *Definição de Processo* foram as entidades que receberam maior ênfase no detalhamento pois elas reúnem a essência do gerenciamento de workflows. As entidades *Cliente*, *Ator*, *Aplicação* e *Serviço de Gerenciamento de Workflows* não receberam uma descrição tão enfática quanto as entidades anteriores. Essas entidades foram implementadas no nível de detalhes suficiente para a participação na validação do sistema, i.e., num cenário real de execução de um processo, descrito no Capítulo 4.

Para cada entidade que foi implementada, segue a descrição dos atributos, relacionamentos, estados, quando necessário, e das operações (parâmetro e valores de retorno). No Anexo A encontra-se o conjunto de interfaces IDL (Interface Definition Language) criado para as entidades.

4.3.1. Linguagem de Definição de Processos e Definição de Processo

A linguagem de definição de processos usada é composta por um conjunto de símbolos (expressões) e significados para estes, que são utilizados para construir as *Definição de Processos*. Estes símbolos devem suportar conceitos que permitam a modelagem das peculiaridades dos processos de negócio existentes no mundo real. Exemplos desta peculiaridades, entre outras, são: (i) execução paralela de tarefas; (ii) execução seqüencial de tarefas; (iii) sincronização de execuções paralelas (AND JOIN); (iv) especificação de papeis.

A linguagem proposta neste trabalho foi construída baseando-se no *modelo de tarefas* proposto pela Nortel. Este modelo mostra a descentralização da coordenação da execução dos processos [13][24], onde a execução é alcançada através de um conjunto de tarefas que conhecem suas interdependências e que reagem a eventos publicados.

Pelo fato desta linguagem ser utilizada diretamente durante a execução do workflow, ela também deve permitir a especificação da associação entre os diversos recursos tecnológicos (IT) e humanos com as atividades que compõem um determinado processo.

A linguagem de definição de processos proposta não é uma linguagem completa e seu poder de expressão é limitado fazendo comparação com as linguagem adotadas comercialmente, e.g., PIF [19], WPDL [4], PSL [5]. Além do mais, ela foi projetada especificamente para a modelagem de processos de negócio em um nível de detalhes tal que a definição de processo produzida possa ser utilizada diretamente pela máquina de

workflows. Nenhum estudo foi feito sobre a adequação desta linguagem para a modelagem de processos diferentes dos processos de negócio.

Os processo de negócio são compostos de tarefas. Tanto o processo quanto as atividades que o constituem, necessitam de um conjunto de dados de entrada, para iniciar sua execução e produzem um conjunto de dados de saída. As tarefas podem ser de dois tipos: (i) tarefas simples e (ii) tarefas compostas. As tarefas simples podem ser definidas como unidades atômicas de trabalho e as tarefas compostas podem ser consideradas como sub-processos, que agrupam logicamente um conjunto de atividades, dentro de outro processo de maior nível hierárquico. Desta maneira, um processo de maior nível hierárquico também pode ser considerado como uma tarefa composta. Além de possuir tarefas, um processo, ou tarefa composta, também possui relações de dependência entre elas, que determinam a seqüência de execução das atividades que o compõem.

A figura 4.4 mostra um processo modelado segundo a notação da proposta da Nortel, seguido da modelagem, do mesmo, utilizando a linguagem de definição de processo proposta, quadro 4.1.

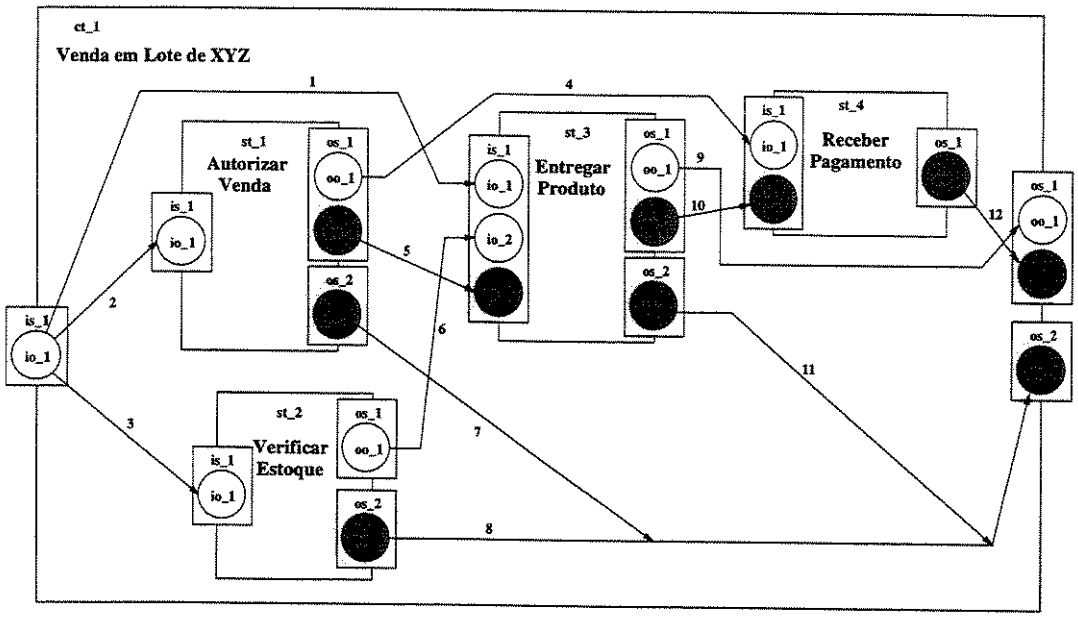

Figura 4.4 - Exemplo de um processo de negócio segundo a notação da Nortel.

```
Compound_Task( ct_1, "Venda em Lote de XYZ" ) {
  Input_Set( is_1) {
    Input_Object( io_1, 'd' ); }
  Output_Set( os_1 ) {
    Output_Object( oo_1, 'd' );
    Output_Object( oo_2, 'n' ); }
  Output_Set( os_2 ) {
    Input_Set( is_1 ) {
      Input_Object( io_1, 'd' );
      Input_Object( io_2, 'd' );
      Input_Object( io_3, 'n' ); }
    Output_Set( os_1 ) {
      Output_Object( oo_1, 'd' );
      Output_Object( oo_2, 'n' ); }
```

```
    Output_Object( oo_1, 'n' ); }
  Composition() {
    Simple_Task( st_1, "Autorizar Venda", "Gerente" ) {
      Input_Set( is_1 ) {
        Input_Object( io_1, 'd' ); }
      Output_Set( os_1 ) {
        Output_Object( oo_1, 'd' );
        Output_Object( oo_2, 'n' ); }
      Output_Set( os_2 ) {
        Output_Object( oo_1, 'n' ); }}
    Simple_Task(st_2, "Verificar Estoque", "Armazenista"){
      Input_Set( is_1 ) {
        Input_Object( io_1, 'd' );}
      Output_Set( os_1 ) {
        Output_Object( oo_1, 'd' ); }
      Output_Set( os_2 ) {
        Output_Object( oo_1, 'n' ); } }
    Simple_Task(st_3, "Entregar Produto", role_type_3 ) {
        Output_Set( os_2 ) {
          Output_Object( oo_1, 'n' ); } }
      Simple_Task(st_4, "Receber Pagamento",
role_type_4 ) {
        Input_Set( is_1 ) {
          Input_Object( io_1, 'd' );
          Input_Object( io_2, 'n' ); }
        Output_Set( os_1 ) {
          Output_Object( oo_1, 'n' ); } }
      Dependency (){
        dep( 1, ct_1.is_1.io_1, st_3.is_1.io_1 );
        dep( 2, ct_1.is_1.io_1, st_1.is_1.io_1 );
        dep( 3, ct_1.is_1.io_1, st_2.is_1.io_2 );
        dep( 4, st_1.os_1.oo_1, st_4.is_1.io_1 );
        dep( 5, st_1.os_1.oo_2, st_3.is_1.io_3 );
        dep( 6, st_2.os_1.oo_1, st_3.is_1.io_2 );
        dep( 7, st_1.os_2.oo_1, ct_1.os_2.oo_1 );
        dep( 8, st_2.os_2.oo_1, ct_1.os_2.oo_1 );
        dep( 9, st_3.os_1.oo_1, ct_1.os_1.oo_1 );
        dep( 10, st_3.os_1.oo_2, st_4.is_1.oo_2 );
        dep( 11, st_3.os_2.oo_1, ct_1.os_2.oo_1 );
        dep( 12, st_4.os_1.oo_1, ct_1.os_1.oo_2 ); }
  }}
```

Quadro 4.1 - Modelagem do processo segundo a linguagem proposta

Descrevem-se, a seguir, os principais elementos da linguagem proposta.

Tarefa Composta (Compound_Task)

Uma tarefa composta pode servir para representar os processos de negócio e/ou para agrupar, em unidades lógicas, várias tarefas que podem ser tratadas como uma tarefa só. Dentro de uma tarefa composta deve haver pelo menos uma tarefa, seja simples ou composta. A sintaxe completa de uma tarefa composta deve possuir: (i) uma ou mais seções para especificar os Input_Sets; (ii) uma ou mais seções para especificar os Output_Sets; (iii) uma seção Composition para especificar quais são as tarefas, sejam simples ou compostas, que a constituem; (iv) uma seção Dependency para especificar a dependência entre as tarefas que foram especificadas no item anterior. A sintaxe de uma tarefa composta está mostrada na tabela 4.1.

```
Compound_Task( id, name ) {
   ...
}
```

Parâmetro	Descrição
id	Identificador unívoco do Compound_Task.

name	Utilizado para se fazer uma breve descrição da tarefa composta.

Tabela 4.1

Tarefa Simples (Simple_Task)

Uma tarefa simples é utilizada para representar uma atividade dentro de um processo. Ela também deve possuir um ou mais Input_Sets e Output_Sets. A sintaxe de uma tarefa simples está mostrada na tabela 4.2.

```
Simple_Task( id, name, role_type ) {
   ...
}
```

Parâmetro	Descrição
id	Identificador unívoco do Simple_Task.
name	Utilizado para se fazer uma breve descrição da tarefa simples.
role_type	Utilizado para definir o papel (role) da entidade que executará a tarefa.

Tabela 4.2

Conjuntos de Entrada (Input_Set) e Conjuntos de Saída (Output_Set)

Os conjuntos de entrada (Input_Sets) e de saída (Output_Sets) são agrupamentos de objetos de dados de entrada (Input_Objects) e objetos de dados de saída (Output_Objects), respectivamente. Estes conjuntos devem possuir pelo menos um objeto de dados especificado dentro do seu escopo. Abaixo estão mostrados as sintaxes dos Input/Output_Sets e também de seus objetos de dados.

```
Input_Set( id ) {
   ...
   Input_Object( id, type );
   ...
}
```

ou

```
Output_Set( id ) {
   ...
   Output_Object( id, type );
   ...
}
```

A definição dos parâmetros tanto dos Inputs quanto Output_Sets são mostradas na tabela 4.3.

Parâmetro	Descrição
id	Identificador unívoco do In/Ouput_Set.

Tabela 4.3

Os objetos de dados, sejam eles de entrada (Input_Object) ou de saída (Output_Object), são utilizados para verificar a satisfação de um determinado conjunto de entrada ou de saída e também para transportar dados entre as tarefas. A sintaxe deles está especificada juntamente com a sintaxe dos conjuntos (acima). Na tabela 4.4 estão os significados dos seus parâmetros.

Parâmetro	Descrição
`id`	Identificador unívoco do In/Output_Object.
`data_type`	O parâmetro `data_type` é utilizado para descrever qual é o tipo do objeto de dados: (i) caracter 'd' para representar que o objeto de dados transporta um conjunto de informações; (ii) a letra 'n' para representar que o objeto de dados transporta uma notificação. Uma notificação é um objeto de dados que não possui conteúdo nenhum.

Tabela 4.4

Composição (Composition)

Serve para descrever as tarefas, simples ou compostas, que uma determinada definição de processo possui, independentemente de suas interdependências. Somente as tarefas compostas possuem uma seção de composição.

```
Composition( ) {
   ...
}
```

Dependência (Dependency)

A seção dependência, que só pertence às tarefas compostas, serve para registrar a dependência entre as tarefas. Cada "dep" dentro da seção Dependency especifica a relação de dependência entre duas tarefas. Na tabela 4.5 estão discriminados os significados dos parâmetros.

```
Dependency( ... ) {
   dep( id, dependee, dependent );
   dep( id, dependee, dependent );
   ...
}
```

Parâmetro	Descrição
`id`	Identificador unívoco da dependência.
`dependee`	Referência absoluta ao objeto de dados do qual se depende (fonte).
`dependent`	Referência absoluta ao objeto de dados dependente (destino).

Tabela 4.5

Entende-se por referência absoluta como sendo um identificador constituído pelo identificador da tarefa, pelo identificador do conjunto e pelo identificador do objeto de dados, separados por pontos, e.g., "simple_task.input_set.input_object".

Resumo

Com base nos elementos apresentados acima, pode-se construir *Definição de Processos* de considerável complexidade.

É essencial que uma linguagem permita a representação de execução paralela e seqüencial de tarefas. Na figura 4.4 são apresentadas ambas as situações. Quando o conjunto de entrada ("ct_1.is_1") da tarefa composta de maior nível hierárquico, i.e., processo "Venda em Lote de XYZ", for satisfeito, as dependências "1", "2" e "3" são satisfeitas. Desta maneira, poder-se-á disparar a execução das tarefas "st_1" e "st_2", evidenciando o paralelismo de execução das duas tarefas. Se as dependências "5" e "6" forem satisfeitas, então a tarefa "st_3" poderá iniciar sua execução.

O termo "satisfeito" é utilizado para designar o estado que um conjunto de entrada ou de saída entra quando todos os seus objetos de dados estiverem sido produzidos.

Este cenário possui duas características importantes. Uma é a execução seqüencial que é mostrada a partir do momento em que a tarefa "st_3" deve ficar inativa enquanto as dependências "5" e "6" não estiverem sido satisfeitas. E a segunda característica é o que se chama de AND JOIN, onde a tarefa "st_3" serve como um ponto de encontro, forçando a sincronização do término de execução das duas tarefas que a precedem.

A figura 4.5 mostra um exemplo genérico de um processo aninhado. A seção Composition da tarefa composta de maior nível hierárquico possui a especificação de outra tarefa composta. No quadro 4.2 encontra-se a *Definição de Processo* gerada para a figura 4.5.

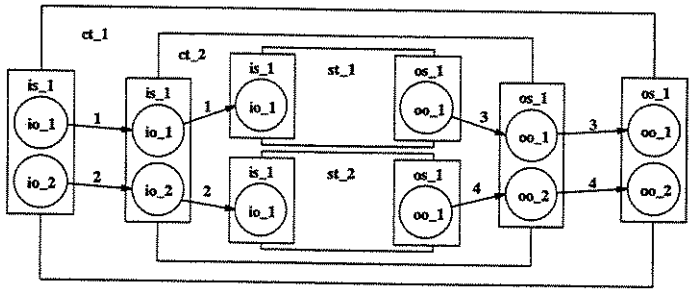

Figura 4.5 - Exemplo de um processo aninhado

Compound_Task(ct_1, any description) { Input_Set(is_1) {	Simple_Task(st_2, any role) { Input_Set(is_1) {

```
        Input_Object( io_1, 'd' );
        Input_Object( io_2, 'd' ); }
    Output_Set( os_1 ) {
        Output_Object( oo_1, 'd' );
        Output_Object( oo_2, 'd' ); }
    Composition() {
        Compound_Task(ct_2, any description) {
            Input_Set( is_1 ) {
                Input_Object( io_1, 'd' );
                Input_Object( io_2, 'd' ); }
            Output_Set( os_1 ) {
                Output_Object( oo_1, 'd' );
                Output_Object( oo_2, 'd' ); }
            Composition() {
                Simple_Task( st_1, "Any",
                                any_role ) {
                    Input_Set( is_1 ) {
                        Input_Object( io_1, 'd' ); }
                    Output_Set( os_1 ) {
                        Output_Object( oo_1, 'd' );
}}
```

```
                    Input_Object( io_1, 'd' ); }
                Output_Set( os_1 ) {
                    Output_Object( oo_1, 'd' ); } }
            Dependency (){
                dep( 1, ct_2.is_1.io_1, st_1.is_1.io_1 );
                dep( 2, ct_2.is_1.io_2, st_2.is_1.io_1 );
                dep( 3, st_1.os_1.oo_1,
                                ct_2.os_1.oo_1 );
                dep( 4, st_2.os_1.oo_1,
                                ct_2.os_1.oo_2 );}}
        Dependency (){
            dep( 1, ct_1.is_1.io_1, st_1.is_1.io_1 );
            dep( 2, ct_1.is_1.io_2, st_1.is_1.io_2 );
            dep( 3, ct_2.os_1.oo_1, ct_1.os_1.oo_1 );
            dep( 4, ct_2.os_1.oo_2, ct_1.os_1.oo_2 );
}}
```

Quadro 4.2 - Modelagem de um processo genérico aninhado

4.3.2. Máquina de Workflows

A *Máquina de Workflows* é a entidade mais importante de um sistema de gerenciamento de workflows. Ela é responsável por interpretar a *Definição de Processo* que lhe foi associada e delegar, conforme a seqüência especificada, a execução das tarefas às entidades executoras competentes. Uma entidade executora é considerada competente para a execução de uma dada tarefa se ela possuir a capacidade de atuar no papel (role) especificado na *Definição de Processo*.

A proposta da Nortel foi a única das propostas submetidas à OMG que apresenta uma padronização para o modelo de coordenação da execução do processo. Este modelo, denominado de *modelo de tarefas* (ver Capítulo 2), foi escolhido para a construção da *Máquina de Workflows* por ser simples e estar disponível publicamente.

A execução de um processo pela *Máquina de Workflows* está dividida em dois passos principais: (i) configuração, onde a *Definição de Processo* é interpretada e são criados os *Controladores de Tarefa* (maiores detalhes sobre esta classe encontra-se mais adiante) que representam as tarefas; (ii) execução propriamente dita, onde os *Controladores de Tarefa* publicam, à medida que resultados são produzidos por eles, e consomem notificações, à medida que os resultados esperados forem produzidos.

A figura 4.6 mostra o diagrama de classes da *Máquina de Workflows* seguido de um detalhamento das classes apresentadas.

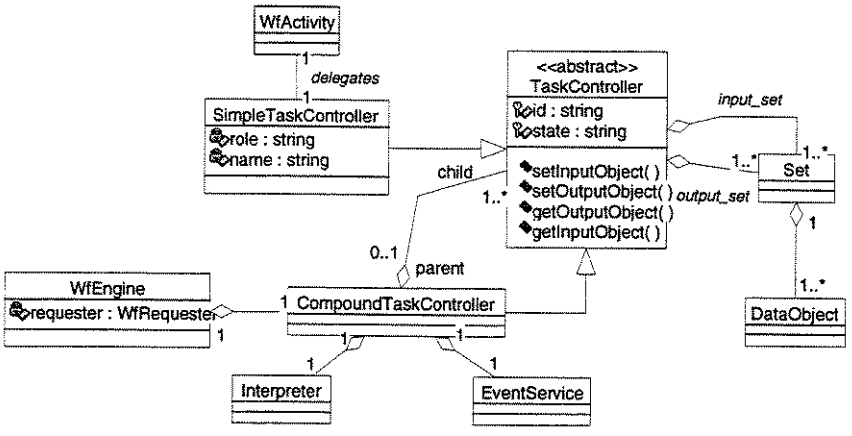

Figura 4.6 - Diagrama de classes da Máquina de Workflows

Controlador de Tarefa (TaskController)

Um *Controlador de Tarefas* é uma classe abstrata que define um conjunto de métodos e atributos para serem reutilizados entre os seus herdeiros. A principal função de um controlador é receber notificações de eventos sobre as dependências que foram satisfeitas e, se estes eventos o habilitar, disparar a execução de determinadas ações. Junto com suas especializações estão descritos os tipos de ações que são tomadas na ocorrência de uma satisfação. Entre os principais métodos que compõem esta classe estão aqueles que permitem definir e recuperar objetos de dados (DataObjects) tanto de conjuntos de entrada (InputSets) quanto de saída (OutputSets).

Na figura 4.7 mostram-se os possíveis estados em que um controlador de tarefas pode estar e na tabela 4.6 encontra-se a descrição destes.

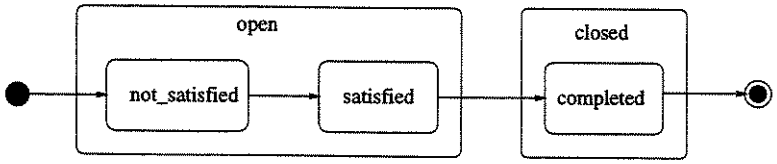

Figura 4.7 - Estados de um Controlador de Tarefa

Estado	Descrição
`opened.not_satisfied`	Um controlador entra neste estado logo após sua iniciação e permanece nele até algum de seus conjuntos de entrada ficar satisfeito.
`opened.satisfied`	Todo controlador entra neste estado quando um e somente

	um de seu(s) conjunto(s) de entrada ficar(em) satisfeito(s). Ele permanecerá nesta situação até que um de seu(s) conjunto(s) de saída seja satisfeito onde passa ao seguinte estado.
`closed.completed`	Todo controlador entra neste estado quando um e somente um de seu(s) conjunto(s) de saída ficar satisfeito.

Tabela 4.6 - Descrição dos estados de um controlador de tarefas

Conjunto (Set)

Um *Conjunto* é a classe utilizada para reproduzir o comportamento apresentado pelos conjuntos de entrada e de saída do modelo da Nortel, armazenando informações dos eventos que são dependentes e daqueles que devem ser propagados quando o conjunto for satisfeito. O conjunto de eventos a propagar é definido com base na seção Dependency da *Definição de Processo* durante o passo de configuração.

Objeto de Dados (DataObject)

Um *Objeto de Dados* é uma unidade elementar de satisfação e transferência de dados entre diferentes *Controladores de Tarefa*.

Controlador de Tarefa Composta (CompoundTaskController)

O *Controlador de Tarefa Composta* é responsável por delegar ao *Interpretador* a criação de todos os seus *Controladores de Tarefa*-filho, sejam eles simples ou compostos, e definir as dependências entre eles de acordo com a *Definição de Processo* que se está executando.

Cada dependência na *Definição de Processo* (*dep(..., ..., ...)*) possui um identificador de dependência (*id*), um identificador absoluto do objeto de origem (*dependee*) e um de destino (*dependent*). Estas informações são utilizadas pelo *Interpretador* para definir as dependências entre os *Controladores de Tarefa*-filho que ele cria. O registro da dependência é feito, tanto no controlador de que se depende, quanto no dependente. Para o controlador dependente (*dependent*), esta informação é útil durante a verificação de sua satisfação mediante a ocorrência de um conjunto de eventos, i.e., conjunto de identificadores de dependência publicados pelo *Serviço de Eventos*. Para o controlador de que se depende (dependee), esta informação é utilizada para compôr o conjunto de eventos que devem ser propagados quando um de seus *Conjuntos de Saída* ficar satisfeito, entrando assim no estado "closed.completed."

O *Controlador de Tarefa Composta* também possui uma entidade, *Serviço de Eventos* (EventService), que provê serviços de notificação assíncrona, toda vez que

ocorrer um evento. Este serviço reproduz, de uma forma simplificada, o mecanismo de notificação do modelo da Nortel. A notificação ocorre, em geral, toda vez que um *Conjunto de Saída* ficar satisfeito.

A tabela 4.7 discrimina as ações tomadas mediante a *satisfação* de um conjunto pertencente ao *Controlador de Tarefa Composta*.

Tipo do Conjunto	Ação
Input	Propaga os eventos para o *Serviço de Eventos* que possui.
Output	Se for o controlador de maior nível hierárquico, envia uma mensagem ao *Máquina de Workflows* o possui notificando do término da execução do processo. Se estiver contido em outro controlador, propaga os eventos para o *Serviço de Eventos* deste último controlador.

Tabela 4.7

Interpretador (Interpreter)

O *Interpretador* é a classe que tem a responsabilidade de ler e interpretar a *Definição de Processo*, criar todos os *Controladores de Tarefa*-filho e utilizar as informações de dependência para estabelecer o relacionamento entre estes. Este serviço é realizado somente durante o *passo de configuração* do *Controlado de Tarefas Composta* de maior nível hierárquico (processo de negócio).

Serviço de Eventos (EventService)

O *Serviço de Eventos* é uma entidade utilizada para propagar, assincronamente, a ocorrência de eventos ao *Controlador de Tarefa Composta* que o contém e todos os seus filhos.

Controlador de Tarefa Simples (SimpleTaskController)

A responsabilidade do *Controlador de Tarefa Simples* é capturar os eventos propagados pelo *Serviço de Eventos*, verificar a habilitação de algum de seus conjuntos de entrada, e se ficar habilitado, instanciar, iniciar e ativar a entidade que realmente realizará o trabalho (*Atividade de Workflow* (WfActivity)). Quando esta produzir os resultados deve-se atribuí-los ao *Conjunto de Saída* adequado.

A tabela 4.8 discrimina as ações tomadas mediante a satisfação de um conjunto pertencente ao *Controlador de Tarefa Simples*.

Tipo do Conjunto	Ação
Input	Instanciar, iniciar e ativar a entidade que realmente realiza o

	trabalho (*Atividade de Workflow*).
Output	Propaga os eventos para o *Serviço de Eventos* do controlador que o possui.

Tabela 4.8

Supondo que o *passo de configuração*, utilizando-se a *Definição de Processo* da figura 4.4, tenha sido realizado, pode-se descrever como a *execução propriamente dita* do workflow é realizada, utilizando-se a *Definição de Processo* da figura 4.4. As premissas anteriores são o conjunto de condições que colocam o *Controlador de Tarefa Composta* ("ct_1) no estado de "opened.not_satisfied."

O *Controlador de Tarefa Composta* de maior nível hierárquico, correspondente ao processo "Venda em Lote de XYZ", somente possui um conjunto de entrada com apenas um objeto de dados, bastando apenas uma atribuição para ficar satisfeito. Quando o controlador entra no estado "opened.satisfied", atribuindo-se valor a este objeto de dados, desencadeia-se a difusão dos eventos ("1", "2" e "3") dependentes da satisfação do conjunto que ficou satisfeito ("ct_1.is_1").

Quando os *Controladores de Tarefa*-filha receberem a notificação, pelo *Serviço de Eventos*, verificam se algum de seus conjuntos de entrada ficou satisfeito em decorrência desta publicação. Se a satisfação ocorrer — como é o caso de "st_1" e "st_2" — um objeto da classe *Atividade de Workflow* é instanciado, para cada *Controlador de Tarefa* satisfeito, seus dados de contexto são definidos com base nos objetos de dados do conjunto de entrada que ficou satisfeito e seu estado é alterado para disparar a sua execução.

Quando a *Atividade de Workflow* entra no estado "opened.running," a primeira ação a ser tomada é encontrar qual é a entidade executora (*Aplicação*, *Ator* ou outro *Máquina de Workflows*) responsável em realizar o papel (role) que se encontra nos dados de contexto. Depois, os dados de contexto da *Atividade de Workflow* (activity_data) devem ser mapeados para esta entidade executora.

A partir do momento em que a *Atividade de Workflow* receber a notificação do término de execução da entidade executora e tiver recuperado os dados produzidos, estes valores são utilizados para definir e satisfazer um dos conjuntos de saída do *Controlador de Tarefa Simples* que requisitou tal serviço.

Diante da satisfação de um conjunto de saída, os eventos que foram registrados como dependentes de sua satisfação são difundidos — eventos "4" e "5" para o controlador "st_1" e "6" para o controlador "st_2" — pelo *Serviço de Eventos*. Desta

maneira, estes eventos poderão causar a satisfação dos demais *Controladores de Tarefa* e a execução continua repetindo estes passos.

Este processo de notificação-habilitação-notificação termina assim que forem publicados os eventos que satisfazerem algum dos conjuntos de saída do *Controlador de tarefa Composta* de maior nível hierárquico — eventos " " e "12" para sat s conjunto de saída "ct_1.os_1" ou, os eventos "8" ou "7" ou "11" para o conjunto de saída "ct_1.os_2").

A discussão feita sobre *Máquina de Workflow*, *Definição de Processo* e linguagem de definição de processos, foi necessária para compreender como é que a execução do processo é alcançada. Está faltando mostrar onde este mecanismo se encaixa nas operações herdadas da interface *Processo de Workflow* (WfProcess). Os parágrafos que seguem fazem a descrição do significado dos atributos, relacionamentos e das operações herdadas.

Atributos

String state - É utilizado para manter o estado do *Máquina de Workflows*. A figura 4.8 mostra os possíveis estado em que esta entidade pode estar e na tabela 4.9 estão descritos os significados destes estados.

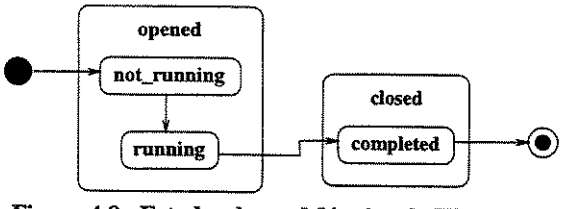

Figura 4.8 - Estados de um Máquina de Workflows

Estado	Descrição
`opened.not_running`	O *Máquina de Workflows* entra neste estado logo após a sua instanciação e permanece nele até uma invocação válida da operação *start()*.
`opened.running`	Neste estado, o *Máquina de Workflows* executa o processo e permanece nele até que o *Controlador de Tarefa Composta* responsável em executar a *Definição de Processo* terminar.
`closed.completed`	A execução do processo termina a partir do momento em que o *Controlador de Tarefa Composta* termina a execução da *Definição de Processo*.

Tabela 4.9

ProcessData context_data - É utilizado para manter os dados de contexto. Estes dados devem ser definidos pelo *Cliente* antes da invocação do método *start()*. A tabela 4.10 mostra a semântica dos elementos de dados contidos na estrutura *context_data*.

Elemento (attribute_name)	Tipo (type_value)	Descrição
process_definition	string	Nome válido de um *Definição de Processo* que será executada pelo *Máquina de Workflows*.
startup_data	ProcessData	Conjunto de dados mínimo que serão processados durante a execução do workflow.

Tabela 4.10

ProcessData result_data - Utilizado para armazenar os dados produzidos após o término da execução do workflow.

Relacionamentos

Client requester - Referência utilizada para enviar notificações, ao requisitante (*Cliente*), sobre as mudanças de estado ocorridas no *Máquina de Workflows*.

Operações

ProcessData result() - Utilizada pelo *Cliente* para recuperar os dados produzidos pela execução do workflow. Esta operação somente estará disponível se o *Máquina de Workflows* estiver no estado "closed.completed."

void set_context(in ProcessData new_value) - O *Cliente* utiliza esta operação para definir os dados de contexto da *Máquina de Workflows*. O principal dado que é passado nesta estrutura é a *Definição de Processo* que será executada.

Na tabela 4.11 estão mostradas as semânticas dos dados que são passados no parâmetro *new_value*. O nome do atributo (attribute_name) e o tipo do valor (type_value) são retornados em uma estrutura do tipo ProcessDataInfo, constituindo as meta-informações sobre os dados de contexto e, mais abaixo, dados de resultado da *Máquina de Workflows*.

Elemento	Tipo	Descrição
process_definition	string	Este parâmetro será utilizado para passar a *Definição de Processo* para a *Máquina de*

		Workflows.
`startup_data`	ProcessData	Servirá para passar o conjunto de dados mínimo para iniciar a execução de um determinado workflow.

Tabela 4.11

ProcessData result() - Esta operação é utilizada pelo cliente para recuperar os resultados produzidos pela *Máquina de Workflows*. A tabela 4.12 descreve a estrutura do valor que é retornado e que também é um dos atributos desta entidade.

attribute_name	type_value	description
`result_data`	ProcessData	Esta estrutura servirá para transportar os dados produzidos pela execução de um determinado workflow.

Tabela 4.12

void start() - Trata-se de um operação assíncrona que dispara a execução propriamente dita do *Máquina de Workflows*. Ela somente pode ser invocada se o *Máquina de Workflows* estiver no estado "opened.not_running" e o *context_data* já estiver sido definido com valores válido. Sobre as circunstância anteriores, a invocação é tida como válida.

A figura 4.9 ilustra a seqüência que ocorre durante a progressão da execução do processo da figura 4.4. Este cenário leva em consideração que o *passo de configuração* já tenha ocorrido.

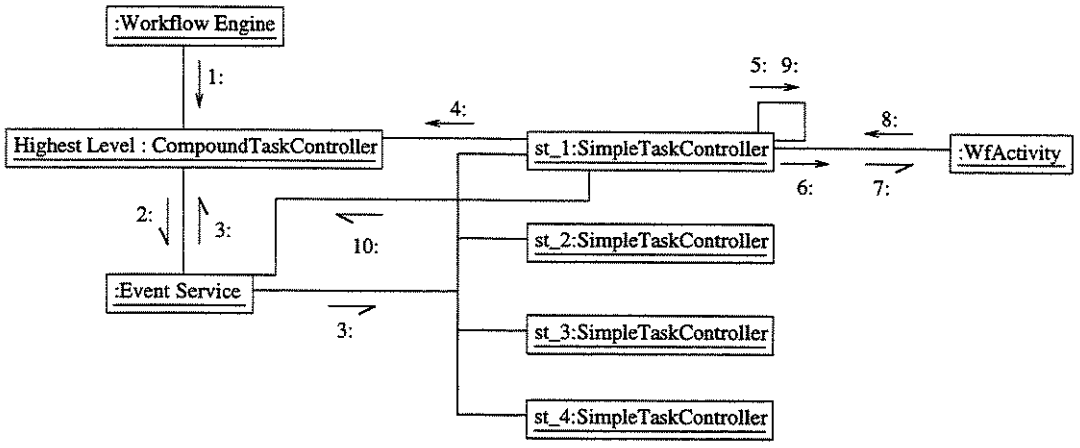

Figura 4.9 - Cenário parcial de execução

1: [`setInputObject( startup_data, "is_1.io_1" )`]. Nesta operação, o *Máquina de Workflows* mapeia o *startup_data*, passado pelo *Cliente*, nos *Objetos de Dados* do conjunto de entrada do *Controlador de Tarefa Composta* de maior nível hierárquico (Highest Level).

2: [`broad_cast([ "1", "2", "3" ])`]. Por ter ficado satisfeito, esta operação é invocada requisitando que o *Serviço de Eventos* propague as dependências satisfeitas "1", "2" e "3."

3: [`receive_event([ "1", "2", "3" ])`]. Os *Controladores de Tarefa*, tanto os filhos quanto o de maior nível hierárquico, recebem, através desta operação, os eventos publicados.

4: [`obj = getInputObject( "is_1.io_1" )`]. Como o *Controlador de Tarefa Simples* "st_1" fica satisfeito com o evento "2", além de outros, ele recupera o *Objeto de dados* disponível.

5: [`setInputObject( "is_1.io_1" )`]. O *Controlador de Tarefa Simples* se atribui o valor acima, entrando no estado "opened.satisfied".

6: [`new(); set_context()`]. Uma *Atividade de Workflow* é criado e seus dados de contexto são definidos com base nos *Objetos de Dados* do conjunto de entrada que ficou satisfeito e no parâmetro *role*.

7: [`change_state( "opened.running" )`]. A *Atividade de Workflow* é ativada.

8: [`complete()`]. A *Atividade de Workflow* após ter delegado a execução a uma entidade executora que desempenha o *role* especificado, devolve os valores produzidos para o *Controlador de Tarefa Simples*.

9: [`setOutputObject( ..., ... )`]. O *Controlador de Tarefa Simples* mapeia os dados produzidos nos *Objetos de Dados* de algum de seu(s) conjunto(s) de saída (s).

10: [`broad_cast([ "4", "5" ])`]. Ao verificar que ficou satisfeito com a atribuição acima, são propagado, através do *Serviço de Eventos*, os eventos (identificadores de dependência) associados ao conjunto de saída satisfeito.

4.3.3. Atividade de Workflow (WfActivity)

A interface *Atividade de Workflow* foi a única interface da jFlow implementada no protótipo. Apesar de seu nome sugerir a realização de uma tarefa dentro de um processo, a sua responsabilidade é de requisitar a execução de uma tarefa, pois ela herda a interface *Requisitante de Workflow* (WfRequester), à uma entidade executora competente para tal, (i.e., *Ator*, *Aplicação* ou *Máquina de Workflow*). Uma entidade executora competente,

como dito anteriormente, é aquela que possui a responsabilidade para atuar no papel exigido pela tarefa, e.g., a tarefa "Autorizar pagamento" só pode ser realizada pela entidade executora que possua o papel "Gerente."

Atributos

String state - Armazena os possíveis estados da *Atividade de Workflow*. Os estados reservados para esta entidade são os mesmos reservados para o *Máquina de Workflows*, diferindo apenas no significado, que está apresentado na tabela 4.13.

Estado	Descrição
`opened.not_running`	Entra neste estado logo após sua instanciação.
`opened.running`	Entra neste estado quando for invocado o método *change_state("opened.running")* e o *context_data* estiver sido definido com valores válidos.
`closed.completed`	Entra neste estado quando a entidade executora associada terminar de realizar o trabalho.

Tabela 4.13

ProcessData context_data - Mantém os dados de contexto da *Atividade de Workflow*. Na tabela 4.14 estão especificados os significados dos elementos que devem estar contidos nesta estrutura.

Elemento	Tipo	Descrição
`activity_role`	string	Nome do papel utilizado para se recuperar a referência à entidade executora competente para a realização do trabalho.
`activity_data`	ProcessData	Dados da atividade que são mapeados para os dados requeridos pela entidade que desempenha o papel acima.

Tabela 4.14

ProcessData result_data - utilizado para manter os dados produzidos pela entidade executora. Na tabela 4.15 está discriminado o significado do elemento contido nesta estrutura.

Elemento	Tipo	Descrição
`activity_result`	ProcessData	Dados produzidos pela execução da entidade executora associada à *Atividade de*

Workflow.

Tabela 4.15

Relacionamentos

SimpleTaskController owner - É utilizado para enviar notificação de término da execução da entidade executora e os valores produzidos por esta ao *Controlador de Tarefa Simples.*

WfResource assignment - Utilizado para manter a referência, quando for o caso, ao *Ator* responsável em realizar o trabalho da tarefa.

WfProcess performer - Utilizado para manter a referência, quando for o caso, à *Aplicação* ou à *Máquina de Workflows* responsável em realizar o trabalho da tarefa.

Operações

void set_result(ProcessData new_value) - Esta operação serve para a entidade executora, que foi associada à atividade, devolver os resultado produzidos.

void complete() - Nesta implementação a *Atividade de Workflow* após receber os dados produzidos pela entidade executora invoca esta operação para terminar sua própria execução.

void change_state(String new_state) - Este método foi a maneira encontrada para ativar a *Atividade de Workflow*, pois na sua interface não existe o método start().

ProcessData context() - Retorna os dados de contexto (*context_data*) da *Atividade de Workflow.*

void set_context(ProcessData new_value) - Utilizado para definir os dados de contexto da *Atividade de Workflow.* O significado dos elementos foram mostrados na tabela 4.15.

ProcessData result() - Utilizado pelo *Controlador de Tarefa Simples* para recuperar os dados produzidos pela atividade (*result_data*), como mostrado na tabela 4.16.

void receive_event(ProcessData event) - Utilizado pela *Máquina de Workflows* ou pela *Aplicação* para enviar as mudanças de seus estados.

A figura 4.10 mostra o diagrama de colaboração em um cenário cuja realização do trabalho se dá automaticamente.

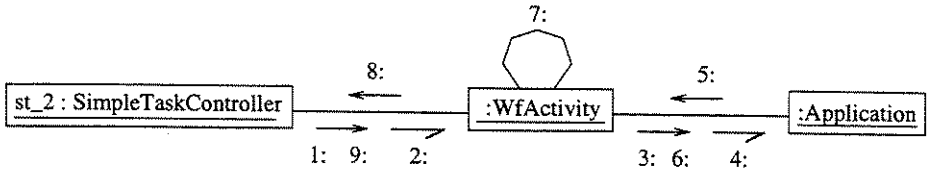

Figura 4.10 - Diagrama de colaboração em cenário automático

1: [`new(); set_context()`]. A *Atividade de Workflow* é criada e seus dados de contexto são definidos baseados nos *Objetos de Dados* do *Controlador de Tarefa Simples*.

2: [`change_state("opened.running")`]. Esta operação ativa a execução da *Atividade de Workflow*. A execução desta operação consiste em encontrar a entidade executora que desempenha o papel requerido e delegar a esta a realização do trabalho.

3: [`set_requester(); set_context()`]. Define o requisitante e os dados de contexto.

4: [`start()`]. Dispara a execução da *Aplicação*. Durante a execução desta operação, os dados de contexto são processados obtendo-se os resultados.

5: [`receive_event()`]. A *Aplicação* notifica o requisitante, *Atividade de Workflow*, do término do trabalho.

6: [`result()`]. A *Atividade de Workflow* recupera os dados produzidos.

7: [`complete()`]. A *Atividade de Workflow* altera seu próprio estado.

8: [`call_back( result_data )`]. Nesta operação o *Controlador de Tarefa Simples* atribui o *result_data* a um conjunto de saída (OutputSet).

A figura 4.11 mostra o diagrama de colaboração em um cenário cuja realização do trabalho se dá semi-automaticamente.

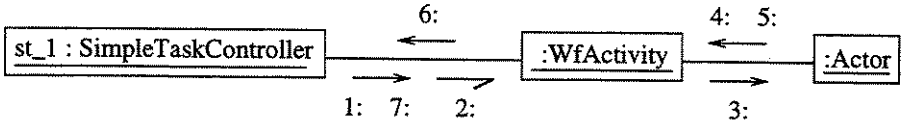

Figura 4.11 - Diagrama de colaboração em cenário semi-automático

As operações 1: e 2: são idênticas àquelas do cenário anterior. A operação 6: deste cenário é igual à operação 8: do cenário anterior.

3: [`add_requester()`]. Com esta operação a *Atividade de Workflow* se registra com o *Ator* que realiza o trabalho.

4: [`get_context()`]. O *Ator* recupera os dados que serão processados por ele.

5: [`set_result(); complete()`]. Após ter processados os dados, o *Ator* devolve os resultados e altera o estado da *Atividade de Workflow*.

4.3.4. Cliente (Client)

O *Cliente* é quem inicia a execução de um processo. Para isso ele deve conhecer o nome da *Definição de Processo* que ele quer executar e o conjunto inicial de dados necessário para ativar o processo requisitado.

Relacionamentos

Vector performer - Esta estrutura mantém a referência a todos os processos requisitados pelo *Cliente*.

Operações

void receive_event(ProcessData event) - Operação utilizada pela *Máquina de Workflows* para informar ao *Cliente* sobre as mudanças significativas de estado da execução do processo requisitado. A tabela 4.16 mostra o significado dos elementos passados na estrutura *event*.

Elemento	Tipo	Descrição
`event_type`	string	Especifica o estado ao qual a *Máquina de Workflows*, cuja referência retornada em `source_process`, encontra-se.
`source_process`	WfEngine	Referência ao *Máquina de Workflows* que gerou o evento.

Tabela 4.16 - Significado dos elementos de *event*

4.3.5. Serviço de Execução de Workflows (Workflow Enactment Service)

A principal função do *Serviço de Execução de Workflows* é a de criar *Máquina de Workflows*.

Relacionamentos

Vector process - Estrutura utilizada para manter a referência a todos os *Máquina de Workflows* que são criados com a operação `create_process()`.

Operações

WfEngine create_process(in WfRequester requester) - Operação utilizada pelo *Cliente* para requisitar a criação de uma *Máquina de Workflows*. O parâmetro passado é a referência ao *Cliente* requisitante.

4.3.6. Aplicação (*Aplicação*)

A *Aplicação* é uma entidade que realiza automaticamente o trabalho de uma atividade. Em um ambiente computacional existirão inúmeras especializações desta entidades, cada qual responsável por um tipo de trabalho. Outra forma de uso desta entidade consiste em construir *wrappers* para os sistemas legados poderem participar da execução de um workflow.

Algumas das operações, atributos e relacionamentos desta entidade são muito semelhantes aos da *Máquina de Workflow*. Desta maneira, somente foram descritas aquelas propriedades que apresentam significados diferentes.

Operações

void set_context(ProcessData new_value) - Utilizado pela *Atividade de Workflow* para definir os dados de contexto que são processados pela *Aplicação*. A tabela 4.17 define o significado do elemento contido dentro desta estrutura.

Elemento	Tipo	Descrição
`startup_data`	ProcessData	Este atributo é uma estrutura utilizada para definir o conjunto mínimo de dados para a iniciação da *Aplicação*.

Tabela 4.17

void start() - Esta operação deve ser especializada pelas diversas variantes da entidade *Aplicação*. Foram implementadas algumas especializações desta operação para a realização da simulação apresentada no Capítulo 5.

4.3.7. Ator (Actor)

Como foi dito anteriormente, um *Ator* serve para executar uma atividade de forma semi-automática. Espera-se que cada pessoa ou grupo de pessoas possua associada a ela uma entidade desta natureza, permitindo-a participar de execuções de workflows.

Operações

void add_request(in WfRequester requester) - Esta operação é utilizada para criar a associação (*assignment*) entre a *Atividade de Workflow* e o *Ator*. Através deste registro

o *Ator* pode obter os dados de contexto, processá-los e devolver os resultados produzidos para a *Atividade de Workflow* que lhe requisitou o trabalho.

Capítulo 5

Cenário de Uso, Aspectos da Implementação e Resultados Obtidos

Na seção 5.1 descreve-se o cenário usado no protótipo implementado e na seção 5.2 são apresentados alguns aspectos e resultados obtidos da implementação do protótipo. É importante ressaltar que os resultados obtidos são só descritivos.

5.1. Execução da Implementação (Cenário de uso)

O *Serviço de Execução de Workflows*, a *Máquina de Workflows* e a *Atividade de Workflow* foram empacotados no mesmo programa (WfEnactmentServiceComponent.class) e instalados em uma máquina Sun rodando o sistema operacional Solaris 2.5.6.

Foram implementados duas variantes da entidade *Aplicação* residindo em dois programas diferentes: (i) AApplicationComponent.class que realiza o trabalho de um "Armazenista", i.e., verifica automaticamente a disponibilidade de produtos no estoque; (ii) BApplicationComponent.class que realiza o trabalho de um "Cobrador", i.e., realiza a cobrança automática do total de um pedido com um cartão de crédito.

Da mesma forma, a entidade *Ator* também foi implementada por dois programas diferentes, AActorComponent.class e BActorComponent.class. O primeiro programa é utilizado para interfacear com a pessoa "Entregador," atribuindo-lhe, quando necessário, a tarefa de entregar uma quantidade de produtos a um determinado cliente. O segundo, é utilizado para interface com a pessoa "Gerente," atribuindo-lhe a tarefa de verificar se o cliente é confiável para realizar a compra.

Cada programa citado acima foi instalado em uma máquina Sun rodando o sistema operacional Solaris 2.5.6. Utilizou-se o Visibroker 3.3 da Inprise como a infra-estrutura de comunicação entre as entidades.

Identificaram-se três etapas durante a experiência de simulação: (i) a escolha de um processo existente no mundo real; (ii) a modelagem deste processo utilizando-se a linguagem de definição de processo proposta; (iii) a execução da definição de processo gerada por um sistema de gerenciamento de workflow em uma possível circunstância real.

O processo escolhido foi a venda em lote de um produto qualquer, que, aqui, será denominado de "Venda em Lote de Produtos." Este processo é composto de quatro atividades, a saber: (i) "Autorizar Venda"; (ii) "Verificar Estoque"; (iii) "Entregar Produto"; (iv) "Receber Pagamento."

Para iniciar o processo, o "Cliente" deve informar os seus dados pessoais, o produto, a quantidade a ser comprada e o número do cartão de crédito utilizado para o pagamento. Fornecido estes dados, as atividades "Verificar Estoque" e "Autorizar Venda" são executadas paralelamente.

Na atividade "Verificar Estoque," o "Armazenista" verifica se a quantidade do produto pedido pelo "Cliente" se encontra no estoque.

Na atividade "Autorizar Venda," o "Gerente" verifica se o "Cliente" possui crédito para a realizar a compra, i.e., se o nome dele está limpo na praça, se sua conta bancária possui os dinheiro para pagar a compra, etc.

Na atividade "Entregar Produto," que é executada a partir do momento que as duas anteriores terminarem com sucesso, o "Entregador" entrega os produtos para o "Cliente."

Na atividade "Receber Pagamento," é debitado do catão de crédito o valor do produtos fornecido ao "Cliente."

O processo termina informando ao "Cliente" que o produto foi encaminhado e que a cobrança foi feita.

Percebe-se nesta descrição que se trata de um cenário bem simples. Porém ele apresenta as características básicas para se testar a adequação da linguagem de definição de processos na modelagem de execução seqüencial e paralela de tarefas.

O quadro 5.1. mostra a definição de processo final resultante da modelagem do processo acima.

```
Compound_Task( ct_1, Venda em Lote de Produtos ) {
 Input_Set( is_1) {
  Input_Object( io_1, 'd');}
 Output_Set( os_1 ) {
  Output_Object( oo_1, 'd');
  Output_Object( oo_2, 'n');}
  Input_Object( io_3, 'n');}
 Output_Set( os_1 ) {
  Output_Object( oo_1, 'd');
  Output_Object( oo_2, 'n');}
 Output_Set( os_2 ) {
  Output_Object( oo_1, 'n');}}
```

```
Output_Set( os_2 ) {
  Output_Object( oo_1, 'n' );}
Process_Definition() {
  Simple_Task( st_1, Autorizar Venda, Gerente ) {
    Input_Set( is_1 ) {
      Input_Object( io_1, 'd' );}
    Output_Set( os_1 ) {
      Output_Object( oo_1, 'd' );
      Output_Object( oo_2, 'n' );}
    Output_Set( os_2 ) {
      Output_Object( oo_1, 'd' );}}
  Simple_Task( st_2, Verificar Estoque, Armazenador ) {
    Input_Set( is_1 ) {
      Input_Object( io_1, 'd' );}
    Output_Set( os_1 ) {
      Output_Object( oo_1, 'd' );}
    Output_Set( os_2 ) {
      Output_Object( oo_1, 'n' );}}
  Simple_Task( st_3, Entregar Produto, Entregador ) {
    Input_Set( is_1 ) {
      Input_Object( io_1, 'd' );
      Input_Object( io_2, 'd' );
  Simple_Task( st_4, Receber Pagamento,
Cobrador ) {
    Input_Set( is_1 ) {
      Input_Object( io_1, 'd' );
      Input_Object( io_2, 'n' );}
    Output_Set( os_1 ) {
      Output_Object( oo_1, 'n' )s;}}
  Dependency (){
    dep( 1, ct_1.is_1.io_1, st_3.is_1.io_1 );
    dep( 2, ct_1.is_1.io_1, st_1.is_1.io_1 );
    dep( 3, ct_1.is_1.io_1, st_2.is_1.io_1 );
    dep( 4, st_1.os_1.oo_1, st_4.is_1.io_1 );
    dep( 5, st_1.os_1.oo_2, st_3.is_1.io_3 );
    dep( 6, st_2.os_1.oo_1, st_3.is_1.io_2 );
    dep( 7, st_1.os_2.oo_1, ct_1.os_2.oo_1 );
    dep( 8, st_2.os_2.oo_1, ct_1.os_2.oo_1 );
    dep( 9, st_3.os_1.oo_1, ct_1.os_1.oo_1 );
    dep( 10, st_3.os_1.oo_2, st_4.is_1.io_2 );
    dep( 11, st_3.os_2.oo_1, ct_1.os_2.oo_1 );
    dep( 12, st_4.os_1.oo_1, ct_1.os_1.oo_2 );}}}
```

Quadro 5.1 - Definição de processo do cenário realizado

A tabela 5.1 mostra a relação das características, quanto a automação, dos programas implementados para representar as entidades executoras que realizam o trabalho das tarefas durante a simulação.

Tarefa	Papel	Automação	Programa
Autorizar Venda	"Gerente"	Semi	BActorComponent
Verificar Estoque	"Armazenista"	Automática	AApplicationComponent
Entregar Produto	"Entregador"	Semi	AActorComponent
Receber Pagamento	"Cobrador"	Automática	BApplicationComponent

Tabela 5.1

A figura 5.1 mostra o momento em que o "Cliente" faz o pedido para a iniciação do processo "Compra em Lote de Produtos" para o *Serviço de Execução de Workflows*. A partir deste momento, uma nova *Máquina de Workflows* é criada para coordenar este processo. Os valores dentro dos colchetes, na janela do *Cliente*, são os dados de contexto passados para a *Máquina de Workflows*.

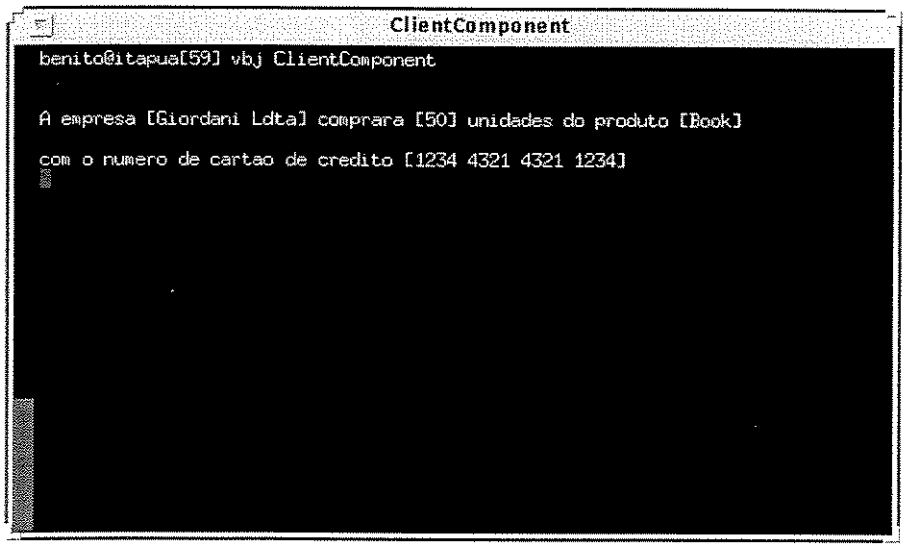

Figura 5.1 - Requisição do processo pelo Cliente

As figura 5.2 e 5.3 mostram as ativações das atividades "Autorizar Venda" e "Verificar Estoque," respectivamente. Por se tratar de uma atividade semi-automática, a primeira tarefa aguarda que a pessoa "Gerente" responda a pergunta para concluir a atividade. Esta interação não ocorre com a segunda atividade que verifica a disponibilidade do produto e termina a tarefa.

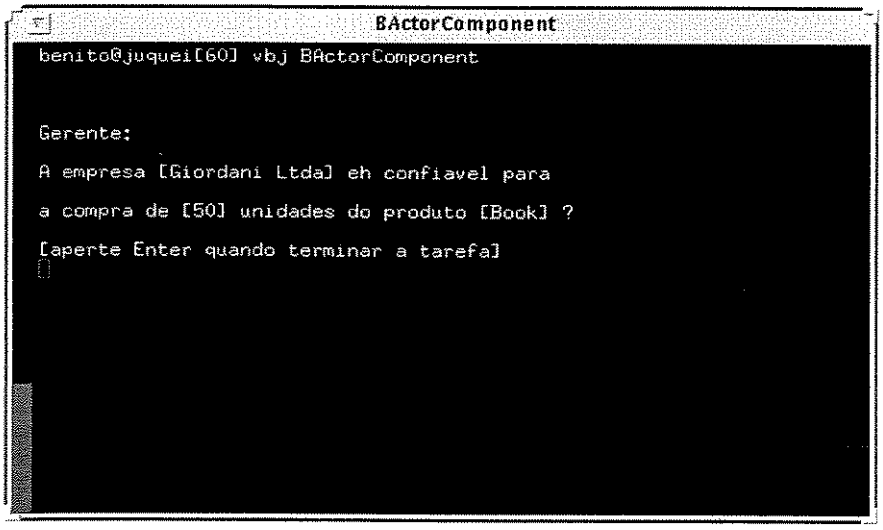

Figura 5.2 - Tarefa "Autorizar Venda"

Figura 5.3 - Tarefa "Verificar Estoque"

Após ambas as tarefas anteriores terem sido completadas, a tarefa "Entregar Produto" é iniciada (figura 5.4)

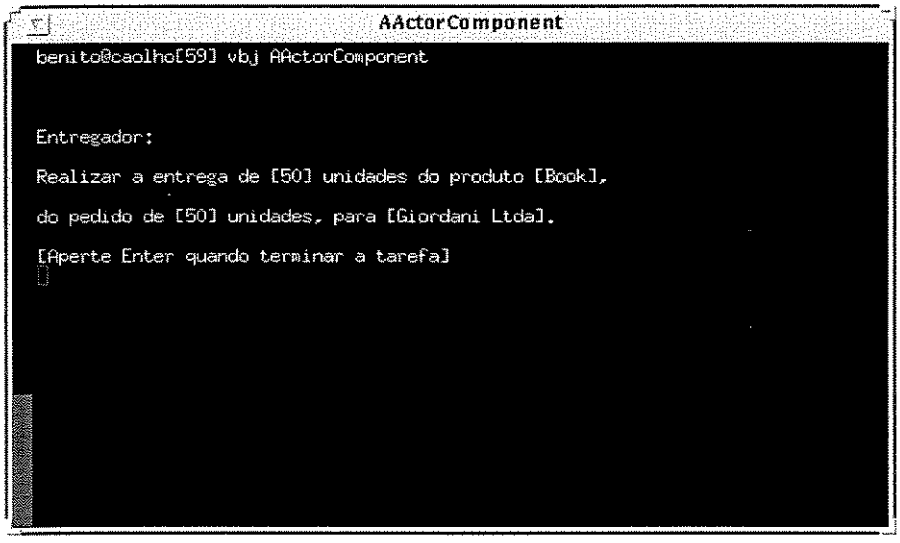

Figura 5.4 - Tarefa "Entregar Produto"

A tarefa "Receber Pagamento" é iniciada quando a tarefa anterior terminar. A figura 5.5 mostra sua ativação.

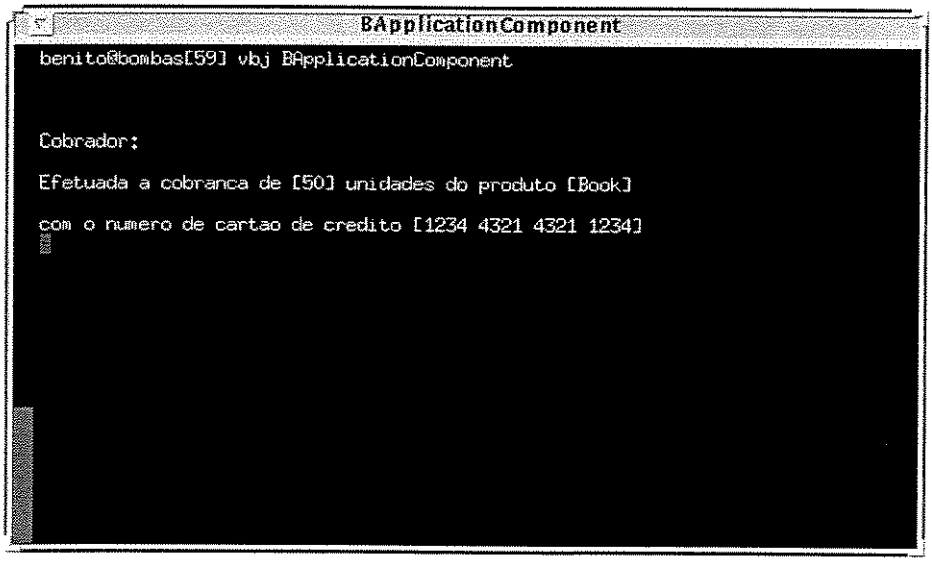

Figura 5.5 - Tarefa "Receber Pagamento"

Quando esta última tarefa terminar, o processo também termina. A figuras 5.6 mostra a notificação para o requisitante, *Cliente*, do término do processo. A figura 5.7 mostra a recuperação dos dados produzidos. A figura 5.8 mostra as mensagens que foram geradas durante o ciclo do processo pelo *Serviço de Execução de Workflows*.

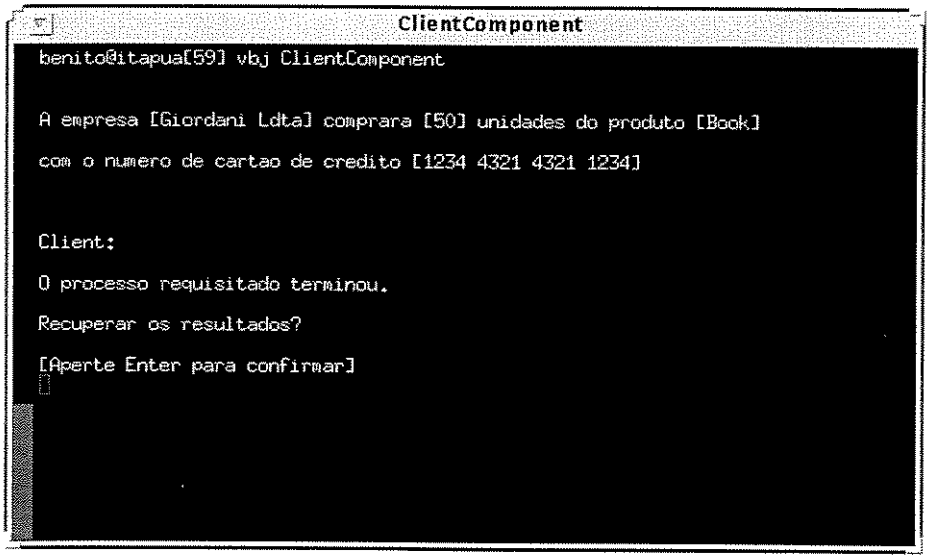

Figura 5.6 - Notificação do término do processo

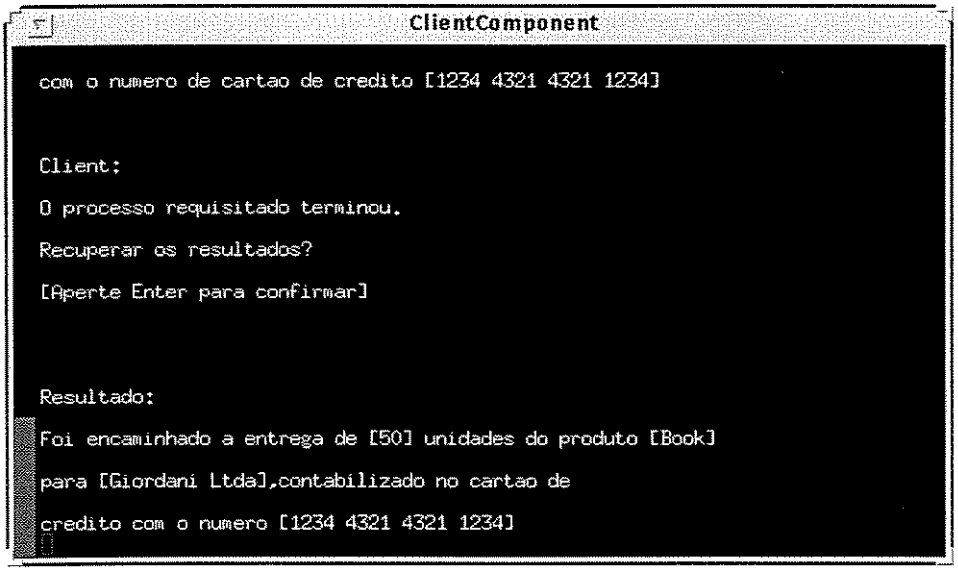

Figura 5.7 - Recuperação dos dados produzidos pelo processo

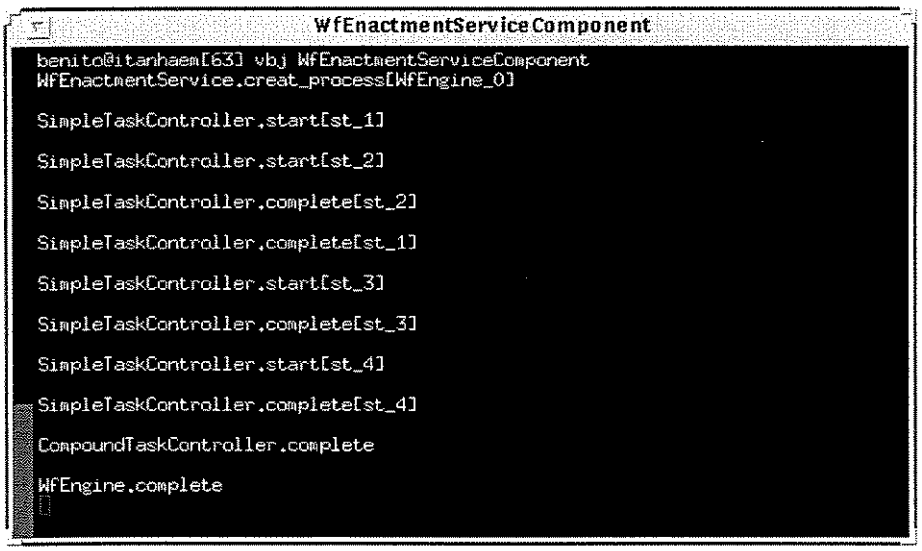

Figura 5.8 - Mensagens geradas durante a execução do workflow

No Apêndice B estão catalogados as semânticas dos dados de contexto e resultados das entidades criadas para a execução do cenário descrito acima.

Prevê-se que uma abordagem semelhante a esta catalogação seja necessária para a tecnologia de componentes, onde os *objetos de negócio* exponham interfaces para a recuperação de suas meta-informações (introspecção). Isto se faz necessário para se estabelecer o relacionamento entre estes componentes para compor uma aplicação específica.

5.2. Problemas de implementação e soluções adotadas

Considerações sobre a distribuição

A figura 5.9 mostra como as classes resultantes das entidades do modelo combinado foram agrupadas em programas distintos e distribuídas em um ambiente computacional utilizando-se a infra-estrutura CORBA. A discussão que se segue tenta fazer um levantamento dos problemas e soluções surgidos em decorrência da decisão de empacotamento adotada. Também são apresentados os motivos que levaram a adoção deste modelo de empacotamento.

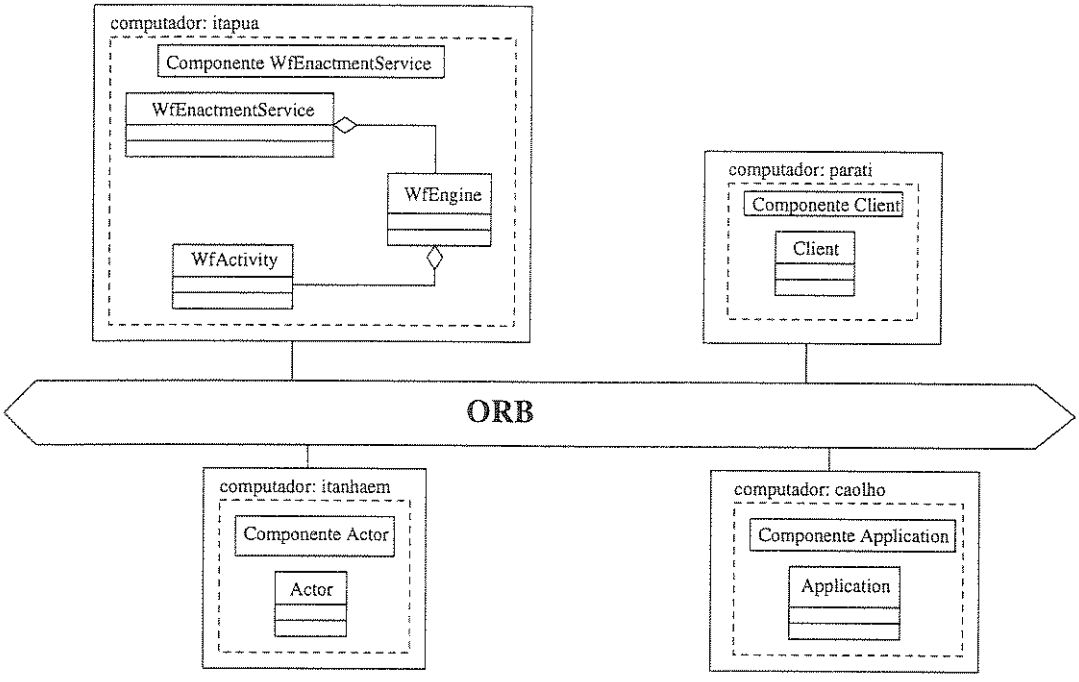

Figura 5.9 - Diagrama de distribuição das entidades

É possível, com a linguagem IDL, qualificar uma operação de uma interface com a expressão *oneway*. Existem algumas regras para se fazer esta qualificação, por exemplo, a operação qualificada deve necessariamente retornar um *void* e não pode haver nos parâmetros desta operação valores passados qualificados com *OUT* (o parâmetro somente é utilizado para retornar um valor quando a operação terminar) ou *IN/OUT* (o parâmetro pode ser utilizado para passar um valor para a operação e pode ser utilizado para retornar um valor quando a operação terminar).

O qualificador *oneway* causa uma espécie de assincronismo na invocação da operação, i.e., o cliente da operação não fica bloqueado aguardando o controle retornar da operação invocada. Este é um recurso muito utilizado para evitar *deadlocks* e usado com freqüência no protótipo principalmente nas operações para alterar o estado (*change_state()*) das *Atividades de Workflow* e da iniciação (*start()*) da *Máquina de Workflows* e das *Aplicações*.

É importante notar que o mecanismo de assincronismo acima funciona quando os objetos, mais especificamente, a execução das operações, é feita em *threads* distintas. Em outras palavras, se vários objetos forem instanciados em um mesmo espaço de endereçamento com apenas uma *thread*, as invocações simultâneas de seus métodos ficam amarradas seqüencialmente (mesmo se a invocação do método for via ORB), impedindo o paralelismo de execução das requisições quando for necessário, em decorrência da natureza do processo modelado.

O problema acima foi observado no programa WfEnactmentServiceComponent.class, pois a entidade *Serviço de Execução de Workflows* pode instanciar diversas *Máquinas de Workflows* e cada um destes podem possuir um conjunto de *Atividades de Workflow* associada, cada qual recebendo invocações externas assíncronas.

Para resolver este problema, as *Máquinas de Workflows*, e suas respectivas *Atividades de Workflow*, possuem sua própria *thread*. Falando em termos de implementação com a linguagem Java, a instância destas entidades é feita dentro de um classe auxiliar que estende a classe Thread.

Uma solução alternativa para o problema acima é instanciar cada uma destas entidades em um espaço de endereçamento próprio, porém, esta alternativa foi descartada devido ao grau de dificuldade inerente à implementação desta solução.

Considerações sobre o uso de CORBA como infra-estrutura de comunicação

O paradigma orientado a objetos distribuídos, da programação inerente à infra-estrutura CORBA, mostrou-se muito próximo ao paradigma orientado a objetos não distribuídos.

Considerações sobre a linguagem de programação escolhida

O protótipo foi construído com a linguagem Java 1.1.7. Foram utilizadas 29 classes que resultaram em aproximadamente 2400 linhas de código. A *biblioteca de classes* (class library) do Java foi de grande utilidade para a implementação do protótipo. Entre as

classes mais utilizadas cita-se: (i) a classe Thread, utilizada para criar novas linhas de execução, necessárias para resolver o problema descrito acima; (ii) a classe StringTokenizer, utilizada para a segmentação de strings e que foi utilizada para a construção da classe *Interpretador*; (iii) possibilidade de executar o protótipo em diferentes sistemas operacionais (portabilidade) apenas alterando a implementação do ORB.

Considerações sobre a linguagem de definição de processo proposta

O objetivo do protótipo não era criar uma linguagem de definição de processos completa. Somente foram propostos os elementos necessários para a validação do sistema de gerenciamento (ver Capítulo 3). Porém, podemos citar algumas extensões que podem ser feitas na linguagem a fim de expandir o seu potencial de expressão. Dentre eles destacam-se: (i) definir o suporte a construções condicionais (e.g., *if*, *while*, *repeat*) que são necessárias para especificar decisões; (ii) total implementação dos tipos de controladores de tarefas previstos no modelo da Nortel; (iii) criar um novo tipo de objeto de dados (In/OutputObject) que permita especificar valores para acesso intermediário (ocorrerem durante a execução da tarefa); (iv) e em termos gerais formalizar a sintaxe da língua.

Considerações sobre o nível de componentização proposto pelo modelo da jFlow

O nível de desacoplamento não é suficiente para uma total independência da implementação dos componentes que correspondem às entidades executoras (objetos de negócio), devido à limitações semânticas e sintáticas na linguagem de definição de processos.

Considerações sobre o controle de concorrência e outros

A implementação não considerou os problemas relacionados à concorrência e à recuperação sob ocorrência de falhas. Espera-se que, num futuro próximo, o serviço de concorrência previsto na infra-estrutura CORBA possa resolver este problema.

Considerações sobre a linguagem notacional utilizada para a construção dos modelos

Foi utilizado o conjunto de notações previsto na UML para a criação dos modelos do protótipo. Das notações utilizadas as que mais se destacam são: (i) o diagrama de interface e de classes que foram utilizados para se especificar os aspectos estáticos do protótipo e o (ii) diagrama de colaboração que apresentou um poder de expressão

consideravelmente alto no que diz respeito a descrição e esclarecimento dos aspectos dinâmicos que as operações do sistema apresentam.

Capítulo 6

Conclusões

Este trabalho permitiu verificar, a partir da implementação do protótipo, a possibilidade da fusão dos modelos apresentados pela jFlow, Nortel e WfMC, em um único e coerente modelo para um sistema de gerenciamento de workflows. Desta maneira, poder-se-ia afirmar que uma proposta conjunta teria sido plausível sem lesar as estruturas propostas por tais grupos.

Criou-se um protótipo simples, porém funcional, de um sistema de gerenciamento de workflows que sustenta a validade do modelo combinado proposto, além de permitir a realização de experimentos futuros.

Através da implementação do protótipo mostrou-se que a infra-estrutura CORBA é adequada para a implementação do sistema de gerenciamento de workflows onde os componentes que o constituem estejam distribuido em diferentes localidades. Através do uso desta infra-estrutura também pode-se concluir que os componentes do sistema de gerenciamento de workflows podem ser implementados com diferentes linguagens de programação e sistemas operacionais.

Constatou-se a necessidade de que cada objeto da classe Máquina de Workflows (Workflow Engine) seja instanciado em um espaço de endereçamento próprio para permitir execuções paralelas, resultantes de requisições provindas de diferentes clientes.

Considerou-se o meta-modelo proposto na UML de fácil entendimento e utilização o que permitiu a criação de modelos (estáticos e dinâmicos) bastante expressivos que facilitaram a compreensão e o desenvolvimento do protótipo nos diversos níveis de abstração em que foram utilizados.

Foi proposta uma linguagem de definição de processos com os elementos básicos necessários para a especificação de processos de negócio que permitiu a implementação do protótipo. Apesar do poder de expressão limitado desta linguagem, ela permitiu a aplicação do sistema de gerenciamento de worflows em um possível cenário real.

Não se conseguiu o total desacoplamento dos componentes que compõem o sistema de gerenciamento de workflows devido à dificuldade em se estabelecer padrões que garantam a interoperabilidade semântica entre os diversos componentes do sistema de gerenciamento de workflows. Tem-se a intuição de que a solução dos problemas relativos à interoperabilidade semântica sejam resolvidas através de suas padronizações.

A linguagem Java, e em particular a classe Thread, apresentou as facilidades necessárias para a implementação dos mecanismos de concorrência do protótipo.

O trabalho apresentado está sujeito a melhorias e deve ser complementado com pesquisas adicionais. Alguns dos assuntos que devem ser considerados para trabalhos futuros são: (i) a implementação da aplicação apresentada numa infra-estrutura de componentes, e.g., Enterprise JavaBeans [3], levando-se em consideração os aspectos relativos à concorrência, segurança e transação; (ii) aumentar o poder de expressão da linguagem de definição de processos proposta para possibilitar o uso de regras mais complexas; (iii) a utilização do protótipo implementado para realizar o levantamento de critérios de avaliação e, com base nestes, realizar comparações entre implementações alternativas de sistemas de gerenciamento de workflows, sendo que uma implementação de grande interesse é aquela onde o mecanismo de coordenação esteja distribuído em diferentes localidades.

Capítulo 7

Referências Bibliográficas

[1] "*Complete CORBAServices Book*," Object Management Group Document, [ftp://.omg.org/pub/docs/formal/98-12-09.pdf].

[2] "*CORBA 2.2 specification-Complete formal CORBA/IIOP 2.2 specification*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/formal/98-07-01.pdf].

[3] "*Enterprise JavaBeans Specification - Version 1.0*," Sun Microsystem Document, [ftp://ftp.javasoft.com/docs/ejb/ejb.10.pdf].

[4] "*Process Definition Interchange V 1.0 Final*," Workflow Management Coalition Document, [http://www.aiim.org/wfmc/standards/docs/if19811r3.pdf].

[5] "*The Process Specification Language*," Process Specification Language Project, [http:// www.mel.nist.gov/psl/index.html].

[6] "*The Workflow Reference Model*," Workflow Management Coalition Document, [http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf].

[7] "*UML Extension for Business Modeling, v1.1*," Object Management Group Document, [ftp.omg.org/pub/docs/ad/97-08-07.pdf].

[8] "*UML Extension for Business Modeling*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/ad/97-08-07.pdf].

[9] "*Workflow Client Application Application Programming Interface (Interface 2 & 3) Specification*," Workflow Management Coalition Document, [http://www.aiim.org/wfmc/standards/docs/if2v20f.pdf].

[10] "*Workflow Management Facility - Request for Proposal*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/ad/97-05-06.pdf].

[11] "*Workflow Management Facility Specification - EDS submition*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/bom/97-08-06.pdf].

[12] "*Workflow Management Facility Specification - jFlow Submission*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf].

[13] "*Workflow Management Facility Specification - Nortel Submission*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/bom/98-01-11.pdf].

[14] "*Workflow Security Considerations*," Workflow Management Coalition Document, [http://www.aiim.org/wfmc/standards/docs/secure1.pdf].

[15] D. Georgakopoulos, M. Hornick, A. Sheth, "*An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*," Distributed and Parallel Databases, Kluwer Academic Publishers, No. 3, 1995.

[16] F. Casati, P. Grefen, B. Pernici, G. Pozzi and G. Sánchez, "*WIDE Workflow Model and Architecture*," Politecnico de Milano, April, 1996.

[17] H. Eriksson, M. Penker, "*UML Toolkit*", John Wiley & Sons, 1997.

[18] I. Jacobson, "*Object-Oriented Software Enginnering - A Use Case Driven Approach*," ACM Press, Addison Wesley, 1992.

[19] J. Lee, M. Gruninger, Y. Jin, et al, "*The PIF Process Interchange Format and Framework*," [http://ccs.mit.edu/pif2.html].

[20] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "*Object-Oriented Modeling and Design*," Prentice Hall, 1991.

[21] K. Riemer, "*A Process-Driven, Event-Based Business Object Model*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/bom/98-11-02.pdf].

[22] M. Fowler, K. Scott, I. Jacobson, "*Uml Distilled : Applying the Standard Object Modeling Language*," Addison-Wesley, 1997.

[23] M. Rusinkiewicz, A. Sheth, "*Specification and Execution of Transactional Workflows*," Modern Database Systems: The Object Model, Interoperability, and Beyond, W. Kim, Ed., ACM Press, 1994.

[24] M. Schimidt, "*Building Workflow Business Objects*," OOPSLA'98 Business Object Workshop IV, http://jeffsutherland.org/oopsla98/mts.html.

[25] N. Krishnakumar, A. Sheth, "*Specification of Workflows with Heterogeneous Tasks in METEOR*," Large Scale Distributed Information Systems Lab, University of Georgia, Athens.

[26] R. Medina-Mora, T. Winograd, and R. Flores, "*ActionWorkflow as the Enterprise Integration Techonology*," Bulletin of the Technical Commitee on Data Engineering, IEEE Computer Society, Vol. 16, No. 2, June, 1993.

[27] S. Das, K. Kochut, J. Miller, A. Sheth, D. Worah, "*ORBWork: A Reliable Distributed CORBA-based Workflow Enactment System for METEOR*$_2$," Large Scale Distributed Information Systems Lab, University of Georgia, Athens.

[28] S. McMenamin, J. Palmer, "*Essential System Analisys*," Prenticel-Hall, 1984.

[29] T. Winograd and R. Flores, "*Understanding Computers and Cognition*," Addison-Wesley, 1987.

[30] W. Schulze, "*Evaluation of the Submissions to the Workflow Management Facility RFP*," Object Management Group Document, [ftp://ftp.omg.org/pub/docs/bom/97-09-02.pdf].

Apêndice A

Neste anexo estão descritos as interface IDLs das entidades geradas para o protótipo.

```
module WfBase {
      struct NameValueInfo{
            string attribute_name;
            string type_name;
      };
      typedef sequence<NameValueInfo> NameValueInfoSequence;

      struct NameValue{
            string the_name;
            any the_value;
      };
      typedef sequence <NameValue> NameValueSequence;

      typedef sequence <string> NameSequence;
};

module WorkflowModel{
      interface WfExecutionObject;
      interface WfProcess;
      interface WfRequester;
      interface WfProcessMgr;
      interface WfActivity;
      interface WfResource;
      interface Client;
      interface WfEnactmentService;
      interface WfEngine;
      interface Application;
      interface ResourceSelector;
      interface Actor;

      typedef WfBase::NameValueInfoSequence ProcessDataInfo;
      typedef WfBase::NameValueSequence ProcessData;

      interface WfRequester {
            long how_many_performer();
            void receive_event( in ProcessData event );
      };
```

```
        interface Client : WfRequester {
        };


        interface WfProcessMgr {
                long how_many_process();
                void set_process_mgr_state( in process_mgr_stateType
new_state );
                ProcessDataInfo context_signature();
                ProcessDataInfo result_signature();
                WfProcess create_process( in WfRequester requester );
        };

        interface WfEnactmentService : WfProcessMgr {
        };

        interface WfExecutionObject {
                string state();
                void change_state( in string new_state );
                ProcessData context();
                void set_context( in ProcessData new_value );
                ProcessData result();
                void resume();
                void suspend();
                void terminate();
                void abort();
        };

        interface WfProcess : WfExecutionObject {
                WfRequester requester();
                void set_requester( in WfRequester new_value);
                WfProcessMgr manager();
                void start();
        };

        interface WfEngine : WfProcess {
        };

        interface Application : WfProcess {
        };

        interface WfActivity : WfExecutionObject, WfRequester{
                WfProcess container();
                void set_result( in ProcessData result );
                void complete();
        };

        interface WfResource {
        };

        interface Actor : WfResource {
                void add_request( in WfRequester );
        };
};
```

Apêndice *B*

Neste apêndice estão descritos as padronizações semânticas das estruturas de dados passadas na comunicação entre as entidades, estabelecidas para criar e realizar a simulação do protótipo do sistema de gerenciamento de workflows.

ProcessData é uma estrutura dinâmica e seqüencial utilizada para mover dados entre as entidades do protótipo. Ela é composta de dois campos: (i) *the_name* - do tipo *string* e utilizada para armazenar o nome da variável que está sendo movida; (ii) *the_value*, do tipo *any* e utilizada para armazenar o valor da variável cujo nome está definida em *the_name*. O tipo *any* suporta qualquer tipo definido no CORBA, seja ele um tipo primitivo, uma estrutura estática ou dinâmica, ou uma referência à uma interface. Desta maneira, a estrutura ProcessData basicamente é utilizada para mover dados entre as entidades, seja através da definição dos dados de contexto ou recuperação dos resultados produzidos por uma entidade, etc.

Porém, o uso do tipo acima causa a necessidade de um acordo entre a entidade que está enviando a estrutura e a que está recebendo. Desta maneira, abaixo está definidos as organizações desta estrutura com relação ao sentido da movimentação dos dados e as entidades que estão envolvidas nessa movimentação.

A tabela 4.1 mostra a estrutura dos dados de contexto do processo "Venda em Lote de Produtos."

Elemento	Tipo	Descrição
`process_definition`	string	Nome da definição de processos. Neste casos "Venda em Lote de Produto."
`startup_data`	ProcessData	Dados do cliente (definidos na tabela 4.2).

Tabela 4.1

Elemento	Tipo	Descrição

client_name	string	Nome do cliente comprador.
product_name	string	Nome do produto a ser comprado.
product_quantity	short	Quantidade do produto a ser comprado.
credit_card_number	string	Cartão de crédito para a cobrança.

Tabela 4.2

A tabela 4.3 mostra a estrutura dos dados resultantes do processo.

Elemento	Tipo	Descrição
client_name	string	Nome do cliente comprador.
product_name	string	Nome do produto comprado
product_quantity	short	Quantidade comprada.
credit_card_number	string	Número do cartão de crédito utilizado na compra

Tabela 4.3

Na atividade "Autorizar Venda" o "Gerente" verifica se o "Cliente" que está requisitando a compra possui o crédito para tal, i.e., se o nome dele está limpo na praça, se sua conta bancária possui os dinheiro para pagar a compra, etc. Os dados de contexto desta atividade são iguais ao startup_data do processo. Já os resultados se encontram na tabela 4.4.

Elemento	Tipo	Descrição
client_name	string	Nome de um cliente confiável.
credit_card_number	string	Cartão de crédito do cliente confiável.

Tabela 4.4

Na atividade "Verificar Estoque", o "Armazenista" verifica se há a quantidade requerida do produto. A tabela 4.5 mostra a estrutura do resultado.

Elemento	Tipo	Descrição
quantity_available	short	Quantidade disponível do produto em estoque.

Tabela 4.5

Na atividade "Entregar Produto", o "Entregador" faz a entrega do produto requisitado ao "Cliente." Os dados de contexto desta tarefa são a soma dos startup_data

do processo, uma notificação da tarefa "Autorizar Venda" e a quantidade disponível do produto pedido em estoque. A tabela 4.6 mostra a estrutura do resultado desta tarefa.

Elemento	Tipo	Descrição
`client_name`	string	Nome do cliente ao qual o produto foi entregue.
`product_name`	string	Nome do produto entregue.
`delivered_quantity`	short	Quantidade do produto entregue.

Tabela 4.6

Na atividade "Receber Pagamento", o "Cobrador" debita o valor da compra do cartão de crédito do "Cliente".

No cenário proposta, o cliente "Giordani Ltda" faz a requisição de 50 unidades do produto "Book" com o cartão de crédito "1234 4321 4321 1234" para o *Serviço de Execução de Workflows* de uma dada empresa que vende este produto.

Apêndice C

Este apêndice é um resumo de um subconjunto da *Linguagem para Modelagem Unificada* (Unified Modeling Language) usada neste trabalho. Por ser um resumo é inevitável a supressão de muitos detalhes. Uma descrição formal de UML pode ser encontrada em [17] e [22], ou pode ser obtida diretamente a partir do *site web* da Rational em www.rational.com.

UML é "uma linguagem para especificar, construir, visualizar, e documentar os artefatos de um sistema de software" [17]. Basicamente consiste de notações gráficas que provêem diferentes perspectivas de um projeto. As notações suportadas pela UML são: (i) diagrama de casos de uso; (ii) diagrama de classe; (iii) diagrama de transição de estado; (iv) diagrama de atividade; (v) diagrama de seqüência; (vi) diagrama de colaboração; (vii) diagrama de componente; (viii) diagrama de distribuição (deployment diagram).

Neste apêndice estão resumidos o diagrama de classes e o diagrama de colaboração por terem sido os diagramas mais freqüentemente utilizados neste trabalho.

Diagrama de Classes

O diagrama de classes da UML permite mostrar a estrutura estática das classes e os relacionamentos entre elas. Um exemplo é mostrado na figura c1.

Classe

Na figura C.1, uma classe é mostrada como um retângulo que possui três partes. A primeira contém o *nome da classe*, a segunda contém os *atributos* e a terceira as *operações*. A visualização de atributos e operações é opcional. A notação permite apresentar outras detalhes da classe, dos atributos e das operações tais como o tipo dos atributos, os parâmetros das operações e o tipo retornado pelas operações.

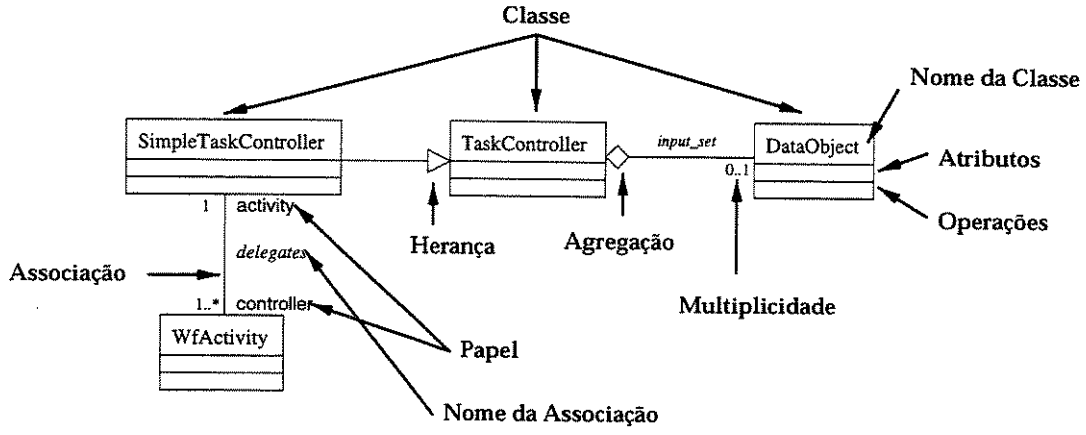

Figura C.1 - Diagrama de classes

Associação

As classes possuem relacionamentos com outras classes (por exemplo, a classe *SimpleTaskController* é uma subclasse da classe *TaskController*), e instâncias possuem relacionamentos com outras instâncias (por exemplo, uma instância de uma *WfActivity* pode ver uma instância de um *SimpleTaskController*). Uma *associação* é um relacionamento entre instâncias. As notações para uma associação estão mostradas na figura C.1.

Uma associação é mostrada como uma linha que une duas classes. Na figura C.1, podemos ver que há uma associação entre uma instância de um *SimpleTaskController* e uma instância de um *WfActivity*.

A notação permite mostrar outros detalhes de uma associação, tais como, o *papel* de cada objeto. O papel pode ser mostrado colocando-se o nome do papel perto da classe relevante. Na figura C.1, o objeto *WfActivity* tem o papel de *controller*, e o objeto *SimpleTaskController* tem o papel de *activity*. Um papel também tem multiplicidade, que indica o número de objetos que podem estar envolvidos em uma associação. Na figura, podemos ver que o objeto *WfActivity* pode ver exatamente um objeto *SimpleTaskController* (indicado pelo 1 próximo a classe *SimpleTaskController*). Lendo-se a associação no outro sentido, podemos ver que um objeto *SimpleTaskController* pode ser visto por um ou mais objetos *WfActivity* (indicado pelo texto "1..*" próximo a classes *WfActivity*. Outra multiplicidade geralmente utilizada é "0..*" (zero ou mais). Finalmente, a associação pode ser nomeada; no exemplo, a associação tem o nome "delegates".

Agregação

Uma *agregação*, um caso especial de uma associação, é utilizado para especifica o relacionamento "parte de um todo," no qual um objeto é considerado o "todo" e o outro objeto é a "parte". A notação para uma agregação é mostrada na figura c.4.

Uma agregação é mostrada como um losângulo acoplado a classe que representa o "todo". A UML suporta uma forma mais forte de agregação conhecida como composição. Na figura C.1, podemos ver que um objeto *TaskController* agrega zero ou mais *DataObjects*.

Generalização

Genericamente dizendo, uma generalização é um relacionamento entre um elemento de modelagem mais geral para um elemento mais específico. Uma generalização é conhecida como *herança*; uma classe (a *superclasse*) age como uma generalização de outra (*subclasse*). A figura C.1. mostra a notação para a herança.

Uma generalização é mostrada como uma linha entre elementos com um triângulo acoplado ao elemento mais geral. Em nosso diagrama, podemos ver que um *TaskController* (o elemento mais genérico) é a superclasse de *SimpleTaskController* (o elemento mais específico) e que *SimpleTaskController* é uma subclasse de *TaskController*.

Diagrama de Colaboração

O diagrama de colaboração da UML é uma notação para mostrar interações entre objetos. Um exemplo anotado esta na figura C.2.

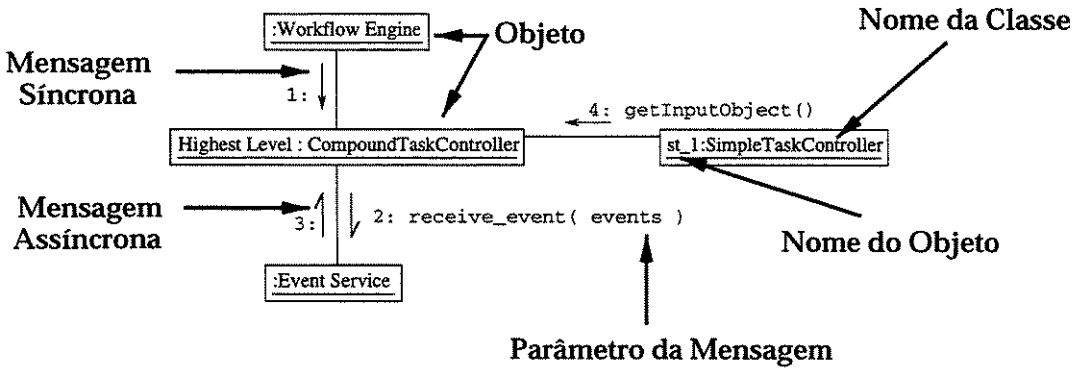

Figura C.2 - Diagrama de Colaboração

Objeto

A notação para um *objeto* é mostrada na figura C.2. Um *objeto* é mostrado como um retângulo que contêm o *nome do objeto* (opcional), dois pontos (:) e o *nome da classe* do qual ele pertence.

Mensagem Síncrona

A notação para uma *mensagem síncrona* é mostrada na figura C.2. Uma mensagem é mostrada como uma seta posicionada próxima a linha contínua entre dois objetos e o seu sentido determina o sentido do envio da mensagem. O número colocado junto a esta seta é utilizado para determinar a seqüência de envio das mensagens. Às vezes o *nome da mensagem* é colocada próxima a esta seta. Os parâmetros da mensagem, que são opcionais, também podem ser mostrados.

No exemplo, o objeto st_1:SimpleTaskController está enviando a mensagem síncrona `getInputObject` para o objeto Highest Level:CompoundTaskController.

Mensagem Assíncrona

A notação para uma *mensagem assíncrona* é mostrada na figura C.2. A notação para uma mensagem assíncrona é idêntica a uma mensagem síncrona, exceto pela ponta de seta de meia cabeça.

No exemplo, o objeto Highest Level:CompoundTaskController está enviando a mensagem assíncrona `receive_event`, que possui o parâmetro `events`, para o objeto sem nome :EventService.