UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FERNANDO MARCOS BONNEMASOU MOREIRA DE CASTILHO

DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE *WORKFLOW* USANDO LÓGICA NEBULOSA

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: PROF. DR. JOÃO BOSCO MANGUEIRA SOBRAL

Florianópolis, Agosto de 2002.

DISTRIBUIÇÃO DE TAREFAS EM SISTEMAS DE *WORKFLOW* USANDO LÓGICA NEBULOSA

FERNANDO MARCOS BONNEMASOU MOREIRA DE CASTILHO

Esta Dissertação foi julgada adequada para a obtenção do título de **Mestre em Ciência da Computação** na área de **Sistema de Computação** e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____
Prof. Dr. Fernando Álvaro Ostumi Gauthier
Coordenador do Curso

Banca Examinadora

_____
Prof. Dr. João Bosco Mangueira Sobral
Orientador

_____
Prof. Dr. Jorge Muniz Barreto

_____
Prof. Dra. Nina Edelweiss

_____
Prof. Dr. Paulo Sérgio da Silva Borges

_____
Prof. Dr. Walter Celso de Lima

“Tudo quanto te vier à mão para fazer,

faze-o conforme as tuas forças...”

Eclesiastes, 9:10.

AGRADECIMENTOS

A meus pais, **Sátyro** e **Rosiris**, a quem resumir a gratidão em uma frase é uma tarefa impossível, meu muito obrigado pelo amor, apoio e incentivo intermináveis.

À **Melissa**, pelo seu amor e sua companhia, que em muitos momentos iluminou o caminho neste trabalho.

Ao Prof. Dr. **João Bosco Mangueira Sobral**, orientador e amigo, pela amizade, ensino e encaminhamento, permitindo superar dificuldades.

Ao Prof. **Tadeu Cruz**, pela prontidão em responder com idéias valiosas, mas principalmente, pela amizade, que me é motivo de honra.

À Profa. **Clorice Pohl Moreira de Castilho**, pelo desafio inicial e torcida durante este trabalho.

Ao Prof. **Cristiano Maciel**, bom amigo e colega da UNIRONDON, pelo apoio na coordenação deste mestrado e no meu começo acadêmico.

Aos Prof. **João Bosco de Barros Freitas**, **João Paulo Delgado Preti**, da UNIRONDON, e Prof. **Bruno Bandeira de LaMônica Freire**, companheiros e amigos, colegas de mestrado, pela troca de idéias, que conduziram a um melhor resultado, mas principalmente, pela presença amiga, que se tornou parte da minha vida.

Aos demais professores do CPGCC – UFSC, em especial, aos Prof. Dr. **Jorge Muniz Barrreto**, **Paulo Sérgio da Silva Borges** e **Ricardo Pereira e Silva**, pelo ensino e pela ajuda, em momentos de decisão.

Ao **Ricardo Couto**, colega da UNIRONDON, pelo apoio em uma etapa importante do trabalho.

À **Direção** da **UNIRONDON** e aos **colegas**, professores, técnicos e alunos, pela compreensão durante períodos de ausência, no decorrer deste trabalho.

DEDICATÓRIA

A minha mãe, **Rosiris**, pelo amor incondicional e pelo incentivo em todos os momentos de minha vida. Este trabalho é uma forma tímida e imperfeita de dizer “obrigado”.

A meu pai, **Sátyro**, pelo exemplo de vida, na busca de autonomia e aperfeiçoamento, baseados em ética, respeito e integridade.

À minha querida **Melissa**, por seu amor, companhia e sonhos compartilhados.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

API	*Application Program Interface*
BPR	*Business Process Reengineering*
CORBA	*Common Object Request Broker Architecture*
CSCW	*Computer-Suported Cooperative Work*
ERP	*Enterprise Resource Planning*
OMG	*Object Management Group*
RNA	Rede Neural Artificial
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured* Query *Language*
TI	Tecnologia da Informação
UML	*Unified Modelling Language*
WAPI	*Workflow Application Programming Interface*
WARIA	*Workflow And Reengineering International Association*
WfMC	*Workflow Management Coalition*
WfMS	*Workflow Management System*
XML	*EXtensible Markup Language*

RESUMO

Em sistemas de *workflow* a escolha do melhor ator para a execução de uma atividade é fator determinante de qualidade e desempenho. Os sistemas de *workflow* atuais atribuem a execução da atividade ao primeiro ator disponível, quando há mais de um, qualificado com o papel funcional requerido pela mesma, o que nem sempre garante que a tarefa seja destinada ao ator mais qualificado.

Este trabalho faz um estudo sobre a atribuição de atividades em sistemas de *workflow*, propondo uma nova abordagem que utiliza inteligência computacional, baseada em lógica nebulosa, para a eleição de atores candidatos à execução da atividade. Esta abordagem, feita a partir da análise de parâmetros de desempenho pessoal na execução de atividades, e do papel funcional que o ator desempenha, valoriza as diferenças de capacitação existentes nas organizações, privilegiando as aptidões e diferenças individuais.

PALAVRAS CHAVE: *Workflow*, lógica nebulosa, fuzzy, processo de negócio, distribuição de atividades.

ABSTRACT

In workflow systems the choice of the best actor for the execution of a task is a determinant factor of quality and performance. Current workflow systems attribute the execution of the task to the first available actor, classified according to the required functional role. This attribution occasionally guarantees that the task is assigned to the most qualified actor.

This work studies the attribution of tasks in workflow systems proposing a new approach that utilizes computational intelligence based on fuzzy logic for the election of candidate actors to the execution of the task. This approach, based on the analysis of parameters of personal performance in the execution of tasks and on the functional role the actor plays, values the differences in skills existing in the organizations privileging the individual aptitudes and differences.

KEY WORDS: Workflow, fuzzy logic, business process, task distribution.

CAPÍTULO 1 - INTRODUÇÃO

Este trabalho versa sobre a definição de um mecanismo para a distribuição de atividades entre os participantes do fluxo de trabalho, que tenham o mesmo papel funcional, em processos controlados por sistemas de *workflow*.

Sistemas de *workflow* desempenham um papel decisivo na modelagem e implementação de processos de negócio, integrando pessoas e recursos de TI – Tecnologia da Informação, na execução de atividades, automatizando o envio de resultados entre atividades, e controlando parâmetros de execução. Com mecanismos automáticos de tomada de decisão, esta tecnologia pode aumentar seu escopo de automatização. Nestes sistemas, atividades são destinadas às pessoas de acordo com o papel funcional que elas exercem na organização. Quando mais de uma pessoa têm o papel exigido por uma atividade, escolher o executor que melhor combine qualidade e produtividade, significa otimizar o trabalho. Contudo, o simples envio de uma atividade ao melhor executor não garante produtividade, se ele estiver sobrecarregado. Deve-se ter informações sobre a carga de trabalho, para distribuir atividades automaticamente, de forma a criar um equilíbrio operacional de trabalho por toda a organização.

1.1 TRABALHOS RELACIONADOS

Em **[Cruz98]** é afirmado que o uso de inteligência artificial nos elementos do *workflow* é a base para um WfMS baseado no conhecimento. Isso representa uma mudança de paradigma, e requer estudos mais aprofundados das mudanças nas regras de negócio, através de agentes inteligentes de *software*. O presente trabalho não aborda a evolução do aprendizado de agentes, mas provê adaptação dinâmica e inteligente do fluxo, em função das mudanças de comportamento dos atores.

Em **[Shar00]** é considerado que a medida de desempenho dos executores de um processo pode representar um fator de motivação para o trabalho. Se o impacto da medição é dirigido de forma a incentivar o trabalho, isto se configura em um forte fator determinante do desempenho. A proposta da determinação automática do executor mais

qualificado para uma atividade, a partir de seu histórico de execução, determina um mecanismo livre de preferências pessoais de gerenciamento, e pode representar um fator de motivação pessoal.

Em **[Alve00]** é proposto um modelo que controla os estados dos executores do *workflow* e a disponibilidade para cumprir atividades, destacando o limite de ocupação do ator. Isto pode tratar pessoas dentro de um padrão geral, sem considerar diferenças momentâneas de comportamento e alterações positivas de produtividade podem ser ignoradas. O presente trabalho é uma contribuição ao trabalho citado, uma vez que determina uma forma de ocupação de participantes do *workflow*, dinamicamente, a partir da adequação das atividades ao seu perfil de desempenho individual, momentâneo e passado, e da qualidade resultante de seu trabalho. Além disso, o número de atividades alocadas a um executor, pode não determinar com precisão o seu grau de ocupação, ao se desconsiderar atividades com maior ou menor grau de dificuldade e prioridade de execução.

Em outro trabalho **[Leym00]**, é afirmado que o maior número de pessoas ao qual uma atividade pode ser destinada é um indício da realização da mesma em menor tempo. Mas isto também pode significar que a atividade seja alocada ao executor menos qualificado, prejudicando a relação entre tempo para executar e qualidade final.

Em **[Mano01]** é argumentado que a análise de dados de histórico, ou o uso de um monitor de *workflow*, são maneiras de verificar o desempenho de atores e a distribuição da carga de trabalho, permitindo a correção e a prevenção de gargalos no fluxo de trabalho. Monitores de processos e monitores de carga de trabalho, em sistemas *workflow*, permitem análise em tempo de execução e atuam sobre um contexto de atividades distribuídas aos atores. A determinação do ator mais preparado para uma atividade, utilizando inteligência artificial, provê uma contribuição à melhoria de desempenho do sistema, uma vez que atua antes da distribuição da carga de trabalho. Há que se ressaltar que a possibilidade de se realocar a carga de trabalho entre atores humanos, ocorre em sistemas onde mais de um ator pode realizar determinada atividade, o que justifica a proposta deste trabalho. Neste mesmo trabalho, é afirmado que seres humanos apresentam tempo de resposta grande e seu trabalho contribui para tornar instâncias de *workflow* como de longa duração. Além disso, eles também são

imprevisíveis, por razões além do controle do sistema de *workflow*, além de terem disponibilidade limitada. Desta forma, para acomodar executores de *workflow*, o núcleo do sistema precisa suportar um mecanismo de invocação adequado a humanos. O autor ainda faz a pergunta: “Seria possível adicionar esta funcionalidade, somente quando necessária, permitindo aos desenvolvedores de *software* empacotar grupos de mecanismos otimizados ?” [Mano01].

A proposta deste trabalho atua sobre os dados de histórico de execuções de atividades, por seres humanos. De forma simplificada, o sistema de *workflow* pode atribuir um executor para a atividade, baseado em parâmetros definidos pelo comportamento do próprio ator. A especificação dinâmica do exercício dos papéis funcionais, de acordo com a disponibilidade imediata do ator, poderia ser um mecanismo para atuar em resposta à pergunta feita por Manolescu.

Em **[Schm99]** é afirmado que o sistema de *workflow* disponibiliza documentos eletrônicos aos participantes do *workflow*, submetendo-os a regras de negócio específicas. Os participantes os acessam no todo, ou em parte, e realizam alterações, de acordo com os papéis que desempenham no processo. Neste trabalho, o documento exerce papel decisivo na determinação da dificuldade da atividade.

Segundo **[Bene00]**, facilidades são requeridas para o exame do histórico de execução de *workflow*, para resolver disputas. Determinar o melhor executor, para uma atividade que pode ser objeto de disputa, é criar uma facilidade que atenda a esta necessidade, como proposto neste trabalho.

De acordo **[Hage99]**, a maioria dos WfMS provê facilidades avançadas para a modelagem de hierarquias, capacidades e responsabilidades funcionais. Esta funcionalidade, junto com um conceito baseado em papéis para a atribuição de trabalho, vem a ser um mecanismo poderoso de balanceamento de trabalho, uma vez que apenas colaboradores humanos têm que estar integrados. Uma vez que este trabalho abrange a classificação de agentes humanos para a execução de atividades, esta dimensão será abordada em maiores detalhes, com a aplicação de inteligência artificial.

1.2 DEFINIÇÃO DO PROBLEMA

Quando uma atividade pode ser atribuída a mais de um participante do *workflow*, o sistema permite que o administrador do fluxo gere um item de trabalho para um executor ou, de forma automática, gera um item para cada ator habilitado. Neste caso, quando alguém assume a execução do item, o sistema elimina os outros itens para a atividade [Leym00]. As duas formas citadas para escolha do executor carecem de análise de dados, que forneçam parâmetros de produtividade e qualidade de execução. O fato de um ator com menos aptidão, ou disponibilidade, poder assumir a execução, faz com que a qualidade final do trabalho possa ser prejudicada. Quando o sistema atende a processos de pequena amplitude, o conhecimento dos comportamentos individuais pode amenizar esta limitação, mas em processos envolvendo diversos setores, com maior qualificação pessoal disponível, faz-se necessário decidir entre um número maior de pessoas, com as quais não se têm contato.

1.3 SOLUÇÃO PROPOSTA

Neste trabalho está sendo proposta a criação de um mecanismo para decidir pela atribuição de uma atividade a um, dentre dois ou mais atores qualificados com o papel funcional requerido pela mesma, a ser invocado por sistemas de *workflow*. Como forma de tornar a distribuição inteligente do fluxo de trabalho, aplicável a um grande número de sistemas, foi abordada a adequação ao padrão da *Workflow Management Coalition* – WfMC, para sistemas de *workflow*.

A proposta é baseada em lógica nebulosa (*fuzzy logic*), e contempla uma aplicação que realiza inferência sobre dados de definição do processo, de histórico de execuções e de carga de trabalho pessoal, elegendo o melhor ator para a tarefa. A aplicação é chamada de WfFuzzy.

A proposta confere dinamismo na manutenção ótima da produtividade do fluxo projetado, sem exigir a intervenção constante do administrador humano do fluxo, para determinação do melhor executor, não inibindo sua atuação, quando necessário.

Neste trabalho, todas as considerações serão feitas à execução de atividades em processos, realizada por seres humanos. Citações à execuções automáticas são apenas complementares.

1.4 ESTADO DA ARTE

***Workflow* e ambiente de Internet** – A Internet permitiu que sistemas de *workflow* fossem disponibilizados para usuários situados por todo o mundo, descentralizando a execução de processos, em atividades de comércio eletrônico. Através da Internet ainda é possível a formação de equipes virtuais de trabalho, coordenando processos inter-empresariais, em integração com sistemas distribuídos de ERP – *Enterprise Resource Planning*.

Integração de *workflow* e *middleware* – "Ponto de encontro no coração da plataforma Cliente-servidor" [Cruz98]. A versatilidade está em permitir a existência de redes multiplataforma, além de operar grandes volumes de dados, de forma a maximizar os recursos utilizados nas redes. Exemplo de funcionalidade, antes só existente em *softwar*es da classe *middleware*, são as funções de *datawarehouse* incorporadas ao *workflow*.

***Workflow* desenvolvido totalmente orientado a objeto** – As características de sistemas de *workflow* são encapsuladas em componentes separados. O núcleo da arquitetura compreende componentes que provêm funcionalidades básicas de WfMS, e componentes adicionais permitem que se implemente aspectos avançados de *workflow*. O desenvolvimento de WfMS pode então ser incrementado, através da possibilidade de se adequar aspectos de *workflow* aos componentes que os implementam, ou mesmo de se desenvolver componentes que adicionem novas características desejadas [Mano01].

WfMS com bases de dados distribuídas – O uso de SGBD distribuído, orientado a objeto, com acesso das aplicações ao conteúdo global do banco de dados, de forma transparente, é uma tecnologia de *software* avançada para WfMS [Cruz98].

Sistemas de *Workflow* que realizam balanceamento de carga – Outra tecnologia sofisticada em WfMS é a de sistemas que realizam balanceamento de carga,

em função da parada de um ou mais servidores, refazendo o roteamento dos fluxos, como é o caso do Lotus Domino 4.5 da IBM [Bene00].

O padrão *JointFlow* – É proposto o padrão *jointFlow*, para um meta modelo de *workflow*, que permite a tradução de um modelo de processo em um *framework* [Schm99]. Ele á a base para o padrão da Facilidade de Gerenciamento de *Workflow*, definido pela OMG, e têm a mesma base de definição do meta modelo de tempo de execução da WfMC, citado no capítulo 3 deste trabalho, mas é mais simples. Esta facilidade de *workflow* adapta o padrão de tempo de execução da WfMC para um ambiente de execução de objetos de negócios, definindo um *framework* e as *interface*s para a implementação de aplicações distribuídas de *workflow*, embutidos na infra-estrutura CORBA, provendo os serviços básicos para aplicações de objetos distribuídos. O padrão habilita a inter-operação de componentes de processos, monitoração de execução de *workflow*s e a associação de componentes com os recursos necessários, para implementação em WfMS existentes. O resultado é um grupo de objetos de negócio de *workflow*, em uma infra-estrutura de tempo de execução de componentes de negócio [Omg98].

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em sete capítulos.

O segundo capítulo apresenta uma revisão bibliográfica, com conceitos importantes para o entendimento de sistemas de *workflow*, explorando características dos sistemas, história das tecnologias correlatas, além de necessidades que conduziram aos atuais sistemas de *workflow*. São feitas referências à proposta deste trabalho, no decorrer da explanação dos conceitos. O estado da arte nesta tecnologia completa o capítulo.

O terceiro capítulo apresenta a *Workflow Management Coalition* (WfMC), o seu modelo de referência, uma arquitetura abstrata de sistema de *workflow* e os padrões de interoperabilidade para esses sistemas. O objetivo é determinar o contexto em que se aplica a solução proposta neste trabalho.

O quarto capítulo aborda paradigmas de inteligência artificial, comentando, após os conceitos fornecidos, os aspectos relevantes para a solução proposta. Ao final do capítulo é feita uma comparação entre os aspectos apresentados e determinada a abordagem para o desenvolvimento do trabalho.

O quinto capítulo apresenta o desenvolvimento do sistema WfFuzzy, que sintetiza a proposta de solução ao problema apresentado. É descrita a metodologia adotada, a especificação das variáveis nebulosas, a modelagem de dados e a especificação orientada a objeto, em UML e, finalmente, o desenvolvimento do sistema WfFuzzy, em linguagem Java. Ao final do capítulo, são traçadas considerações relevantes.

O sexto capítulo fornece os resultados obtidos, com a utilização do sistema WfFuzzy, sobre a definição de processos de negócio simulados, com o objetivo de comprovar o comportamento inteligente da solução e sua aplicação em sistemas de *workflow*.

O sétimo capítulo contém as conclusões, a descrição das dificuldades encontradas no decorrer do trabalho, sua superação e considerações sobre trabalhos futuros.

As Referências Bibliográficas e os Anexos estão no final do trabalho.

O **Anexo A** apresenta o modelo de dados do sistema. O **Anexo B** mostra os diagramas de caso de uso e de seqüência, da solução proposta. O diagrama de classes do sistema desenvolvido é exibido no **Anexo C**. Finalmente, nos **Anexos D** e **E**, há três exemplos de códigos em Java, das classes mais importantes do sistema.

CAPÍTULO 2 - WORKFLOW

Neste capítulo, são descritos os elementos básicos para o entendimento de *workflow* e da tecnologia de sistemas de gerenciamento de *workflow* (WfMS). Para tanto, é feita a contextualização deste paradigma, conceituando os elementos da gerência de processos realizada por WfMS.

Workflow

> "Fluxo de controle e informação em um processo de negócio" [Cruz98];

> "Automação de um processo, no todo ou em parte, durante a qual documentos, informação ou tarefas são passadas de um participante a outro para ação, de acordo com um grupo de regras de procedimento" [Holl01].

> "Um *workflow* representa o aspecto operacional de um procedimento de trabalho, a estrutura de tarefas e as aplicações e pessoas que as executam, a ordem da invocação de tarefas, a sincronização e o fluxo de informação que suportam as tarefas, e os mecanismos de busca e disponibilização de informações, que medem e controlam as tarefas" [Mano01].

2.1 SISTEMAS GERENCIADORES DE *WORKFLOW*

Sistemas de *workflow* realizam o controle do fluxo de informação entre um ou mais grupos de trabalho, onde participantes realizam atividades complementares, de forma corporativa. Um sistema deste tipo permite que se estabeleçam os caminhos para os fluxos de trabalho, as regras para as atividades e os participantes que irão desempenhar papéis em processos, com direitos e deveres sobre o acesso e a geração da informação necessária ao trabalho. Estes elementos compõem o cenário do trabalho

envolvendo processos de negócio, e um sistema de *workflow* proporciona uma ferramenta de trabalho em grupo, para definição, controle e execução dos mesmos.

A modelagem de um sistema de *workflow* consiste em encontrar uma solução para as questões relativas a **o que** deve ser feito (processos e *workflows*), **como** (atividades), **por quem** (atores participantes) e **com que recursos** (ferramentas) [Grae97]. A seguir, são fornecidas algumas definições para o entendimento da terminologia adotada no texto.

Sistema Gerenciador de *Workflow*

> "Um sistema que define, cria e gerencia a execução de *workflow*s através do uso de *software*, executando um ou mais motores de *workflow*, capaz de interpretar a definição do processo, de interagir com participantes do *workflow* e, quando requerido, invocar o uso de ferramentas e aplicações de TI" [Holl01].

Motor do *Workflow*

> "Integra todos os serviços disponíveis no *software*, através da *interface* para ativação de ferramentas específicas para a execução de uma determinada atividade e do manipulador das listas de trabalho" [Cruz98].

Manipulador da Lista de Trabalho

> "Um componente de *software* que gerencia a interação entre o usuário e a lista de trabalho, mantida por um motor de *workflow*" [Holl95].

As listas de trabalho são explicadas na seção 2.3.1.

2.2 PROCESSOS DE NEGÓCIO

> "Processos de negócio são conjuntos de atividades, ou procedimentos, interligados, que coletivamente explicitam um objetivo de negócio ou

> meta política, normalmente no contexto de uma estrutura organizacional, definindo papéis e relacionamentos" [Holl01].

> "Um processo pode se estender por mais de uma organização e definir ambientes virtuais, operando como se fossem estabelecidos na estrutura formal da organização. Em essência, os processos definem fluxos de trabalho, onde são elicitadas as rotas por onde devem seguir as trocas de informações, o papel desempenhado pelos participantes e as regras de atuação" [Leym00].

Processos de negócio e *workflow*

Um processo de negócio deve ter uma definição clara dos parâmetros e dos resultados de execução e deve ser gerenciado por um administrador que, através de ferramentas de monitoramento e intervenção, possa garantir sua correta execução. Processos cuja execução não é feita utilizando tecnologia de apoio ao trabalho em grupo, dependem da iniciativa dos executores das atividades, apresentando um limite na eficiência, fixado por um grau considerável de incerteza na definição de parâmetros para a melhor administração de uma cadeia produtiva.

Uma vez modelado um processo, o WfMS permite que ele seja definido em termos de seqüências de atividades, executores, ferramentas e tempo, para então armazenar esta definição. Quando o processo é colocado em produção, ele é instanciado pelo WfMS, e sua definição é requerida pelo sistema, para que as funções do fluxo de trabalho e as de supervisão sejam executadas. Este cenário compõe a base de um WfMS, voltado a atender processos estruturados.

Definição do Processo

> "É a representação de um processo de negócio, em uma forma que suporta manipulação automatizada" [Holl01].

A definição de um processo em um sistema de *workflow*, consiste na representação das atividades, especificando os elementos presentes na sua execução, como participantes, ferramentas de TI e critérios para início e fim de cada atividade. A repetição contínua destas atividades é feita segundo um padrão chamado **modelo de processo**, que varia de acordo com o tipo de negócio de cada organização [Leym00]. O modelo de processo engloba todos os caminhos e atividades a serem executadas, quando uma instância do processo é criada. Os caminhos de execução das atividades determinam os fluxos de execução de trabalho. As atividades podem ser executadas automaticamente, no todo ou em parte, fazendo parte do escopo de gerenciamento do WfMS, ou manualmente, ficando fora deste escopo. As atividades que são executadas em computador pertencem ao modelo de *workflow*. Atividades são conceituadas na seção 2.3.

Na Figura 2.1 é mostrado que, da mesma forma como o modelo de processos instancia processos, o modelo de *workflow* instancia *workflow*s.

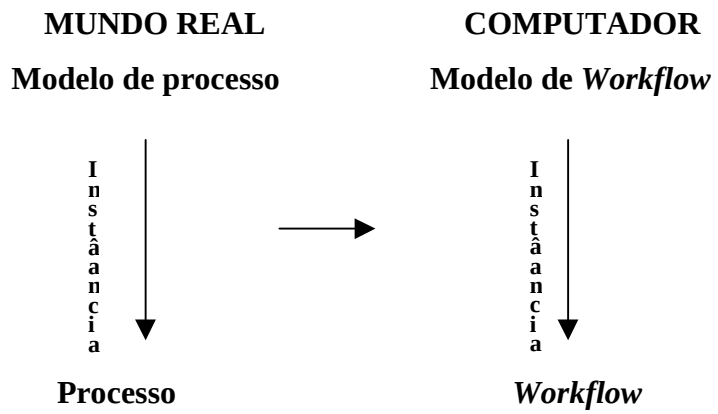

Figura 2.1 – Processos e *Workflow*s. [Leym00]

Quando um processo é chamado, o WfMS gera uma instância do mesmo, que consiste na representação de uma execução do processo. Na instância é representado o estado interno, que especifica os valores dos dados tratados e a identificação do processo, que serve para o armazenamento e recuperação, em operações de auditoria. A Figura 2.2 mostra como processos podem ser invocados por outros processos.

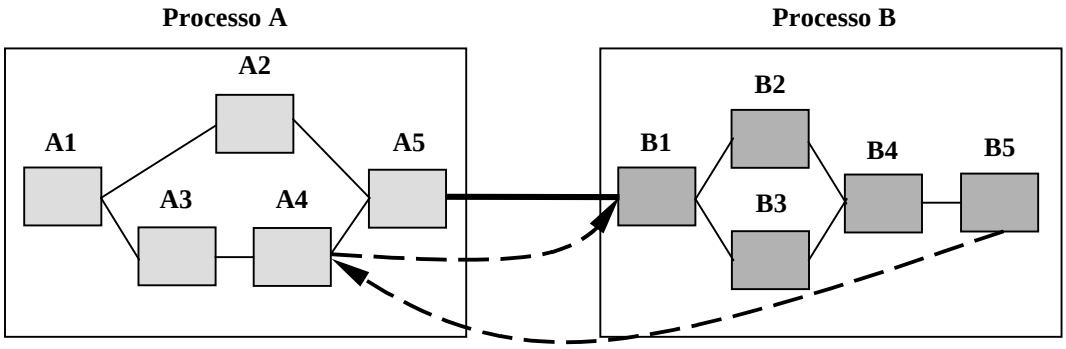

Figura 2.2 – Processos e Atividades. [Holl95]

Processos e *workflow*s podem ser representados através de três dimensões relacionadas: **Processo Lógico**, **Organização** e **Infraestrutura de TI**, conforme Figura 2.3 [Leym00]. O processo lógico determina **o que** deve ser feito, a Organização, **quem faz** e a Infraestrutura de TI, **quais recursos** serão usados para a execução.

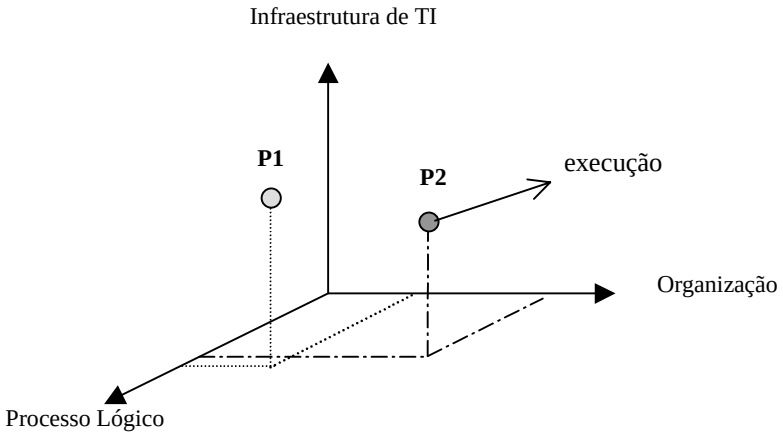

Figura 2.3 – Dimensões de Processos de Negócio. [Leym00]

2.2.1 ESTADOS DOS PROCESSOS

O ciclo de vida de um processo está baseado nas transições de estado que uma instância individual do mesmo pode experimentar, desde sua criação até seu término, normal ou forçado. Estas transições podem ser provocadas pela ação do motor do WfMS, com base em condições estabelecidas na definição do processo, como resultado de eventos externos, ou a partir de intervenções do administrador do *workflow*, ou de outro usuário autorizado, via comandos da *interface* do programa de aplicação de *workflow* (WAPI). As transições são resultado da atuação do serviço de execução do *workflow*, comentado no capítulo 3. Sob este aspecto, o serviço de execução pode ser considerado como uma máquina de transição de estados [Holl95].

A Figura 2.4 mostra as transições básicas de estado que podem ocorrer durante a execução do processo. Cada estado é considerado a seguir, segundo [Leym00].

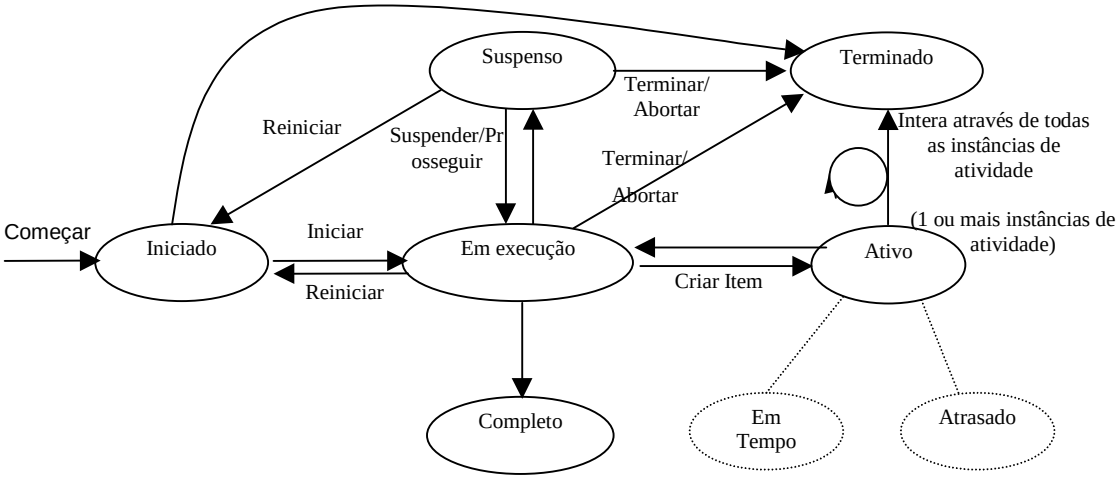

Figura 2.4 – Transições de Estado para Instâncias de um Processo [Holl95]

Inicializado – Ocorre a criação de uma instância do processo. Neste estado são associados apenas dados de estado e dados necessários à execução do *workflow*, como definição de atividades do processo e de participantes do *workflow*. As atividades não são associadas aos executores e o processo pode ser visto em consultas que indicam não terem sido satisfeitas as condições para início de execução.

Em Execução – O processo é carregado e começa a execução de sua instância. Qualquer de suas atividades pode começar, quando uma condição pré-estabelecida para início ocorrer.

Ativo – Neste estado, uma ou mais de suas atividades foi iniciada, através da criação de um item de trabalho, associado a uma instância de atividade.

Em Tempo – A instância do processo está **Ativa** e o tempo determinado pelos tempos das atividades em seqüência, ainda não ultrapassou o ponto de atraso da conclusão do processo como um todo.

Atrasado – A instância do processo está **Ativa** e o tempo limite para conclusão, determinado pelos tempos limite das atividades em seqüência, foi atingido. Neste estado, o sistema deve emitir aviso à administração do *workflow*, e deve ser possível alterar a prioridade das atividades restantes no processo. Esta alteração pode ser feita de forma autônoma, com mecanismos de inteligência artificial, mas não é objeto de consideração neste trabalho.

Suspenso – O processo é colocado neste estado, através de requisição do usuário. A instância do processo está inativa no momento, nenhuma atividade é iniciada e itens de trabalho não são criados. A execução é interrompida, até que o processo volte ao estado de **em execução**, através de outra requisição do usuário, ou devido à ocorrência de um tempo limite para reinício.

Completado – O processo atinge o fim da execução. A instância do processo preencheu as condições de finalização, operações internas, tais como estatística sobre dados, serão executadas e a instância será destruída.

Terminado – A instância do processo foi tirada do estado de execução, por uma requisição do usuário, ou automaticamente, ao atingir uma data limite para término. Neste caso, a execução foi interrompida antes da finalização normal e operações para resumo de erros ou recuperação de dados do resumo serão executadas, e a instância do processo será destruída.

2.3 ATIVIDADES

Atividade é uma unidade de trabalho, a ser executada para um caso ou instância de um processo. Pode ser automatizada, sendo executada sob o controle de um WfMS, ou ter tratamento manual, precisando ser associada a um executor humano. A atividade precisa ser estruturada, de modo a espelhar os elementos necessários à sua execução, que pode ocorrer através da participação única de pessoas, de aplicações computacionais, ou de ambas, em cooperação.

Atividade

> "A descrição de uma unidade de trabalho, que forma um passo lógico em um processo." [Holl01].

2.3.1 ESTRUTURA DE UMA ATIVIDADE

Como pode ser observado na Figura 2.5, o elemento principal da atividade é uma abstração da **Tarefa**, ou o que de fato deve ser feito, relativamente ao negócio. Esta abstração consiste, em **pesquisa na base de dados** da estrutura organizacional, para determinar quem pode executá-la, e em uma **implementação apropriada**, que envolve os recursos para a execução e a forma como estes serão utilizados. O WfMS garante que a atividade seja alocada a apenas um usuário, que pode ser um indivíduo ou uma equipe.

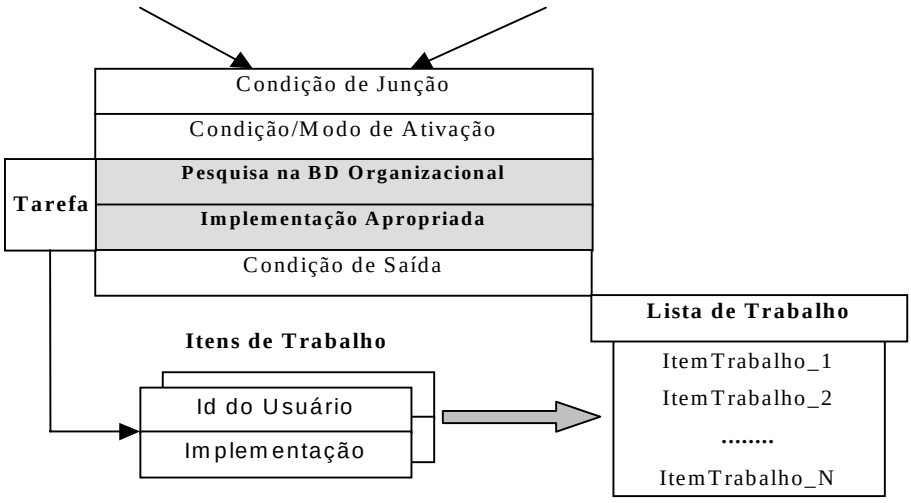

Figura 2.5 – Estrutura das Atividades

Outro componente da atividade é a **condição de junção**, que determina quando ela pode começar, a partir do término de execução de um grupo de atividades anteriores.

A **condição de ativação** determina quando ela deve ser ativada, momento em que a mesma deve ser atribuída à pessoa que irá executá-la. O processo de encontrar as pessoas com o papel funcional exigido pela atividade é chamado de **resolução de quadro funcional**. Quando a atividade está pronta para processamento, é realizada uma pesquisa na estrutura organizacional, no ambiente do *workflow*, que retorna um grupo de usuários aptos para a execução. Os dados organizacionais podem ser invocados a partir de uma aplicação externa ao WfMS, ou a partir do próprio WfMS. No primeiro caso, a aplicação do usuário assume as operações para que seja retornada ao WfMS uma lista de executores habilitados para a atividade. A invocação de dados a partir do WfMS, pode envolver o mapeamento direto entre os meta modelos do WfMS e de dados da organização, ou o acesso direto, quando o modelo de dados do WfMS engloba os dados organizacionais. O **modo de ativação** define se o início da atividade deve ser feito manualmente, ou automaticamente, assim que a **condição de ativação** é atingida [Leym00].

Uma atividade, associada a um executor, passa a ser um **item de trabalho**.

Item de Trabalho

> "A representação do trabalho a ser processado por um participante do *workflow*, no contexto de uma atividade em uma instância do processo" [Holl01].

> "Conjunto de instruções que permitem a um participante do processo de negócio produzir parte do produto final neste processo" [Cruz01].

Os itens de trabalho podem ser executados através do WfMS, ou manualmente, fora do escopo do WfMS, havendo para isto, a necessidade do executor informar ao sistema o resultado do trabalho.

Quando o início da atividade se dá por forma manual, ele é levado a efeito através da interação do usuário com a **lista de trabalho**,

Lista de Trabalho

> "Uma lista de itens de trabalho, associados a um determinado participante do *workflow*" [Holl01].

As listas de trabalho agrupam itens com características comuns, como identificação do ator, ou do papel funcional, requerido para a execução de uma tarefa.

A **condição de saída** determina se a atividade foi completamente executada e se a navegação pelo processo pode continuar. Se a atividade for encerrada e esta condição não tiver sido satisfeita, os itens de trabalho são devolvidos à lista de trabalho, para execução posterior. Desta forma, ocorre o controle do sucesso do trabalho. Como o WfMS trabalha com contexto de execução, o trabalho pode ser continuado a partir do ponto em que parou, a menos que tenha caráter transacional. Neste caso, quando ocorre uma interrupção na execução, operações são desfeitas e o trabalho deve ser reiniciado, de um ponto especificado na transação. A atividade têm a si associada, a representação de seu estado interno, bem como sua identificação, para fins de auditoria. O estado das atividades e do processo estão em estruturas chamadas de **depósitos de entrada**, que armazenam dados para execução de processos, e **depósitos de saída**, que armazenam o resultado da execução das atividades. Sempre que o WfMS precisar saber o estado de um processo ele fará uso destes depósitos. Assim, a execução de atividades pode ser continuada, caso tenha sido interrompida [Leym00].

2.3.2 EXECUÇÃO DAS ATIVIDADES

Quando um processo é instanciado, são instanciadas as atividades, na ordem em que devem ser executadas, os itens de trabalho são gerados para os participantes e a navegação no processo começa. Sempre que a instância de processo invoca uma atividade, é criada uma instância da mesma, associada àquela única instância de processo [Leym00]. As atividades podem ser criadas no momento em que sua execução é requerida, na geração dos itens de trabalho [Hags00].

Atividades passam o controle de execução entre si, invocam programas, e instanciam outros processos. O resultado de seu trabalho é utilizado para o início de outras atividades, representando informação de transição entre elas. Se a atividade for a última a ser executada no processo, seu resultado representa a informação de saída do processo.

A atividade pode ter execução manual, ou automática. Sua implementação pode ser executada na estação de trabalho, no computador que hospeda o WfMS, ou em ambos, representando uma aplicação cliente-servidor [Leym00]. A atividade pode ser executada em interação com o usuário, ou enviando a este o resultado da execução. Ela pode ainda executar operações de consulta e manutenção de base de dados e repassá-los a outras atividades.

As atividades são invocadas para execução de forma automática ou manual, dependendo do momento em que são chamadas. Na Figura 2.6 são mostradas as formas de execução citadas.

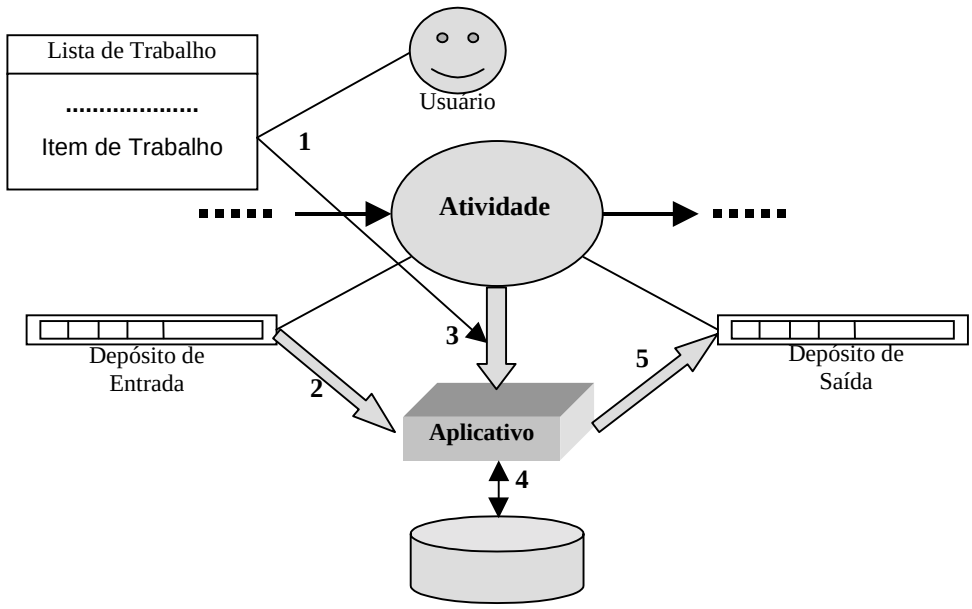

Figura 2.6 – Execução manual de atividade [Leym00]

Em **chamada automática**, a atividade é colocada em ação, de acordo com a definição do processo e sem intervenção humana. Em **invocação manual**, o usuário (1) escolhe um item de trabalho da lista, e a atividade começa a ser executada, em uma seqüência determinada pelo WfMS. Primeiramente, (2) é preenchido o depósito de entrada, com os dados de definição da atividade, que são então substituídos pelos parâmetros da instância da atividade. Em seguida, o programa responsável pelo tratamento das informações (3) é invocado. A partir dos dados do depósito de entrada (2), é executado o processamento da atividade, podendo haver acesso a uma base de dados (4), ou a invocação de aplicativos externos ao WfMS. Ao final desta etapa, o depósito de saída é gravado pelo aplicativo (5), com os resultados da execução da atividade, e o controle é devolvido ao WfMS. Finalmente, a condição de saída é testada. Se não for satisfeita, o item é colocado novamente na lista de trabalho, para que o ator possa continuar a executá-lo posteriormente. A execução deve ser feita dentro de um tempo limite, especificado na definição do processo, com o objetivo do cumprimento de metas de produtividade, ou em função de limitações legais estabelecidas.

2.3.3 ESTADOS DAS ATIVIDADES

Assim como ocorre com os processos, as instâncias de atividades também sofrem transições de estado, que determinam seu ciclo de vida. A Figura 2.7 mostra os estados de uma atividade.

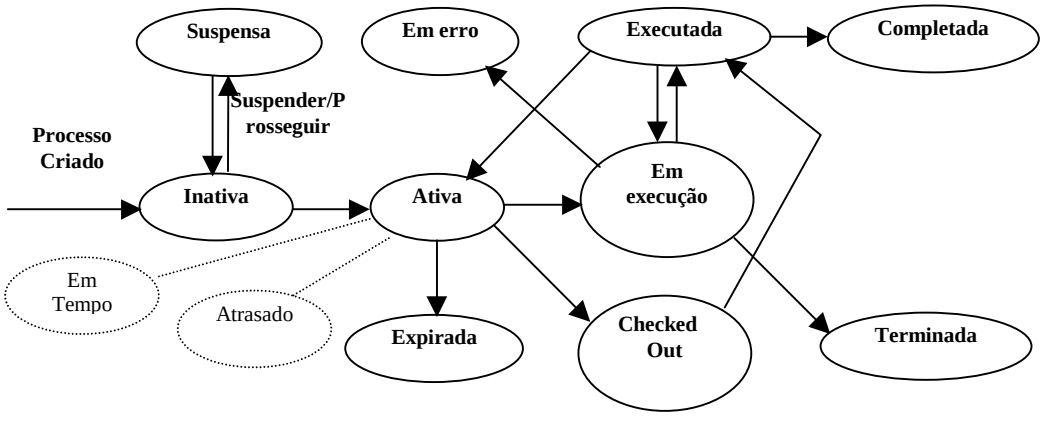

Figura 2.7 – Transições de estado para instâncias de atividades [Leym00]

Os estados das atividades e suas transições, conformes vistos na figura, são descritos a seguir.

Inativa – Uma instância de atividade foi criada para a instância do processo, mas ainda não foi ativada, porque condições de início não foram atingidas. Neste estado, a atividade não apresenta item de trabalho.

Ativa – Este estado indica a criação de um item de trabalho, assinalado para processamento na instância da atividade.

Em Execução – A implementação da atividade é colocada em execução, através de invocação do WfMS. Enquanto a implementação não terminar, a instância permanece neste estado.

Executada – A implementação da atividade terminou e ela atingiu uma condição de saída.

Completada – Uma condição de saída foi atingida, de forma automática, ou de forma manual, quando requerida pelo executor que iniciou a atividade. As condições de transição posterior, serão avaliadas, para que a definição do processo permita a invocação da próxima atividade.

Suspensa – A instância da atividade é colocada em estado de inatividade, através de um comando de alteração de estado. Isto acontece, devido a uma falha em uma condição de junção, ou em um caminho de execução.

Em Tempo – A instância da atividade está em execução, e o tempo limite para término não foi atingido.

Atrasada – A instância da atividade está em execução, e o tempo limite para término foi atingido. Nesta condição, o sistema têm como disparar avisos, e uma aplicação pode, de forma inteligente, alterar a prioridade da atividade, ou tomar outra medida, baseada numa política estabelecida.

Expirada – Quando, por alguma razão, a atividade deixar ser necessária, ela é colocada neste estado, para que a navegação no processo possa continuar.

CheckedOut – A instância da atividade é colocada em execução, através de um comando do usuário, invocando a sua implementação fora do controle do WfMS, com o apoio de uma facilidade de *checkOut*. Neste estado, o usuário pode iniciar a implementação da atividade a qualquer momento, através de uma API apropriada. Quando a execução termina, a atividade vai para o estado de **Executada**, como se o WfMS tivesse realizado a implementação da mesma.

Em Erro – Quando a implementação não acontece da maneira correta, ou não pode ser encontrada, a atividade entra neste estado, de onde sai através de nova tentativa de execução, com ou sem a correção do erro.

Terminada – As atividades podem ser terminadas, individualmente, da mesma forma que os processos. Quando um processo é terminado, as atividades em execução também o são.

Além destas transições de estado, algumas vezes pode ser que funções de suspensão, de reinicio, e de término de atividades necessitem ser executadas, requerendo que todas as atividades em estado de execução tenham que ser completadas. Há ainda situações em que atividades devem ser executadas em bloco, ou em que o processo têm que ser desfeito, até atingir um ponto de reinicio [Holl95].

2.3.4 RELACIONAMENTOS ENTRE AS ATIVIDADES

É necessário que se garanta que um processo será completado, e que as informações resultantes das atividades atingirão seu destino [Cruz98]. Um processo pode percorrer caminhos compostos pela combinação de diversos tipos de rota. A relação entre uma atividade **(T)**, sua(s) antecedente(s) **(A)** e conseqüente(s) **(C)**, define a rota por onde segue o fluxo de execução de um processo. As rotas são estabelecidas na definição do processo, e podem ser do tipo Seqüencial, Divisão, Junção e Repetição. Para Junção e Divisão, dependendo das condições de transição, a relação pode ser do tipo booleano *AND* (E), *OR* (OU) e *XOR* (OU Exclusivo).

Em roteamento seqüencial, como na Figura 2.8, diversas atividades são executadas em seqüência, sob uma única unidade de execução. A conclusão da atividade anterior habilita o início da seguinte.

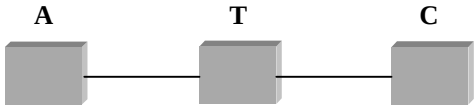

Figura 2.8 – Roteamento Seqüencial

Quando um ponto do *workflow* faz com que uma simples unidade de execução de controle, seja dividida em múltiplas unidades, executadas em paralelo, conforme Figura 2.9, tem-se uma divisão do tipo Divisão-E, que representa um ponto de sincronização, onde uma atividade concluída habilita a alocação de duas ou mais atividades em paralelo. Após o término da atividade que precede a junção, todas as sucessoras são alocadas.

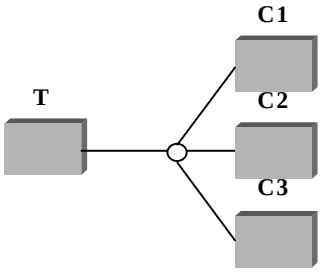

Figura 2.9 – Roteamento com Divisão-E

Se, após a execução de uma atividade, o fluxo deve seguir por mais de uma rota disponível, como na Figura 2.10, dependendo de uma condição de avaliação, ocorre um ponto de divisão do tipo Divisão-OU. Neste caso, a unidade de execução de controle decide para qual das rotas seguir, baseada nos dados avaliados pelas das condições de transição das atividades. Após o término da atividade antecessora, pelo menos uma das sucessoras é alocada. Se for possível seguir por apenas uma rota, o ponto de divisão que se forma é do tipo exclusivo, ou Divisão-XOU.

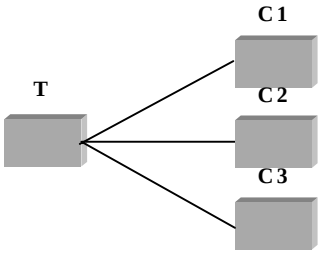

Figura 2.10 – Roteamento com Divisão-OU

Quando duas ou mais atividades, em execução paralela, convergirem para uma única unidade de execução de controle, ocorre um ponto do tipo Junção-E, como na Figura 2.11. Esta convergência estabelece uma sincronização de atividades, na qual cada unidade de execução que termina é mantida em espera, até que todas possam ser convertidas em uma única, que causa o início da próxima atividade. A atividade T só pode ser ativada após o término de todas as antecedentes.

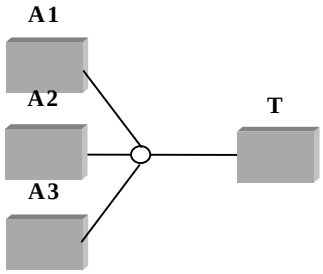

Figura 2.11 – Roteamento com Junção-E

Se não for necessária uma sincronização entre duas ou mais atividades em paralelo, que convergem para um ponto no *workflow*, a junção decorrente é do tipo Junção-OU. A atividade T só pode iniciar, após o término de pelo menos uma das antecedentes. A Figura 2.12 mostra esta situação, caracterizada pelo caráter alternativo das atividades antecedentes.

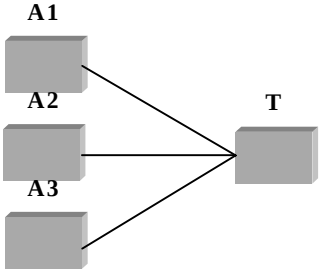

Figura 2.12 – Roteamento com Junção-OU

Quando uma condição de transição exigir a execução repetitiva de uma atividade, ou grupo de atividades, até que elas sejam avaliadas como verdadeiras, conforme Figura 2.13, ocorre um ciclo de *workflow*, ou um roteamento do tipo Repetição, entre as atividades.

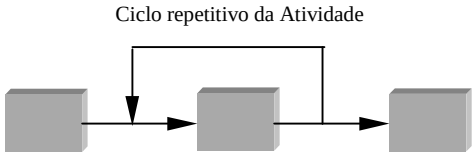

Figura 2.13 – Roteamento tipo Repetição

2.4 PARTICIPANTE DO *WORKFLOW*

Pode representar um ser humano ou um recurso de TI, sendo também chamado de **ator**.

Ator do *Workflow*

> "Um recurso que executa o trabalho representado por uma instância de atividade de *workflow*" [Holl01].

O ator pode representar um **papel funcional**, ou ainda uma **unidade organizacional**. É estabelecido na definição do processo, e participa de instâncias de processos, realizando atividades que lhe são associadas. Se for um **ser humano**, é alguém específico, a quem são atribuídas atividades. No caso de um **recurso de TI**, é um participante de uma atividade totalmente automatizada, podendo ser representado por um agente de *software*. Quando for um **papel funcional**, ou uma **unidade organizacional**, é representado por uma função, ou cargo, no organograma da organização onde o WfMS executa o processo. Se uma **equipe** com vários componentes for um ator, o responsável pela equipe é o seu representante no sistema, recebendo a denominação de super ator e cabe a ele a determinação da execução das atividades do processo, que lhe são atribuídas.

O estado dos atores pode depender de sua condição na estrutura organizacional, ou do grau de ocupação no WfMS.

2.5 RECURSOS

Os recursos necessários à execução de atividades, são *software*, *hardware* e tempo, entre outros. Quando se tratar de *software*, os recursos são aplicativos invocados durante a execução da atividade, como editor de texto, planilha eletrônica, sistema corporativo, ou um sistema legado. Sistemas legados são os existentes na organização, quando da implantação de uma nova metodologia de trabalho, e que não são, completa ou totalmente, integrados ao novo cenário, mas necessitam continuar em operação para suprir determinada necessidade. Neste contexto, recebem especial atenção e demandam esforço para que suas informações façam parte do sistema corporativo da organização.

2.6 ELEMENTOS DA ESTRUTURA DE UM WfMS

Um WfMS deve ser estruturado, de forma a tornar automatizado o fluxo de trabalho coberto pelo escopo de implantação do sistema, conferindo aumento de desempenho, segurança e padronização na execução das atividades dos processos.

Os elementos funcionais do sistema são os papéis, as regras, as rotas, o tempo e os eventos.

Papéis – Na implantação de um sistema de *workflow*, os papéis desempenhados pelos atores definem atributos para situar o fluxo projetado na estrutura organizacional a que ele atende. Há dois tipos de papel. O **papel usuário** identifica as pessoas participantes do fluxo de trabalho e as suas responsabilidades, permite acesso às atividades que devem executar, e define os resultados esperados. O **papel função** especifica ações a serem executadas, sobre quais elementos de dados, e os resultados esperados desta execução. Os papéis usuário e função são associados, definindo as responsabilidades no fluxo [Cruz98]. Neste trabalho, o papel funcional, considerado para a atribuição da atividade ao ator, constitui apenas uma referência à sua função, executada no processo.

Regras – As regras, especificadas na definição do processo, estabelecem o comportamento do fluxo de informações, e os direitos e deveres dos atores, conferindo a autonomia necessária ao desempenho do processo, refletindo com precisão as regras do processo de negócio da organização. Os atributos das regras são os parâmetros de identificação, condições de início e término do processo, tempo para a execução da atividade, atividades anterior e posterior no fluxo, ferramentas para execução, regras de notificação de eventos, condições de segurança e eventos registrados para auditoria [Cruz98].

Rotas – As rotas especificam o caminho de execução estabelecido na definição do processo, e têm um papel funcional, definido pelos papéis funcionais dos atores participantes do fluxo na rota. Para que uma decisão de roteamento possa ser feita entre duas ou mais rotas paralelas, é necessário que elas tenham o mesmo papel funcional [Cruz01]. Os tipos de rota foram detalhados na seção 2.3.4.

Tempo – É a base da definição de produtividade em *workflow*, visando sincronizar o fluxo de trabalho de equipes distribuídas, cooperando em um processo. A aferição da atividade dos executores e da qualidade das métricas de tempo, estabelecidas na definição do processo, pode ser realizada através da comparação entre o tempo projetado para cada atividade e o que de fato foi demandado em sua execução. Neste trabalho, o tempo é um atributo usado em inferências, para a determinação da produtividade do ator. As considerações sobre tempo são feitas no capítulo 5.

2.7 INFRA-ESTRUTURA PARA WfMS

Sistemas de *workflow* são construídos sobre a arquitetura cliente-servidor, com grandes volumes de transações sobre bases de dados corporativas. Na parte cliente, um WfMS é dividido nos módulos de ferramentas para desenho do fluxo de trabalho, de ferramentas para ativação dos objetos modelados, e de ferramentas para roteamento e verificação do *workflow*. A parte servidor é composta pelos módulos de gerenciamento dos serviços de *workflow*, de bancos de dados, de transporte de mensagens e de comunicações. O gerenciamento dos serviços de *workflow* controla todas as funcionalidades do sistema, como ativação de modelos, roteamento, verificação e manutenção de grupos, de usuários, de regras e de rotas. O gerenciamento de serviços

de bancos de dados garante a integridade das bases de dados, em transações multiusuárias. O controle de transporte de mensagens e comunicações gerencia a troca de mensagens entre os participantes do fluxo, e a comunicação destes com o ambiente externo [Cruz98].

Há dois tipos de transações em sistemas de *workflow*, as globais e as de negócio. Transações globais envolvem grupos de atividades executadas, sem a interação de usuários. Estas permitem a recuperação automática do estado do sistema, em caso de falhas durante a execução, e são totalmente executadas em ciclos medidos em segundos. Transações de negócio, por sua vez, são processadas através da interação com o usuário do sistema, com execução medida em minutos ou horas, e podem envolver sistemas diferentes, efetuando suas próprias transações, independentes de seus pares. O conjunto de atividades componentes de transações é chamado de esfera atômica. A atomicidade é garantida por transações de compensação dos efeitos das transações originais sobre as outras, quando as primeiras têm que ser desfeitas [Leym00]. Sistemas de *workflow* gerenciam transações, e compreendem em sua modelagem, os limites das transações globais, ou de negócio, bem como das atividades de compensação.

A plataforma cliente-servidor é a que melhor têm atendido aos sistemas de *workflow*, com uma estrutura de três camadas.

Camada de comunicação com o usuário – Executa no computador cliente as aplicações que fazem uso das API, para acesso ao servidor do WfMS. Nesta camada, a GUI (*Graphical User Interface)* fornece o ambiente de interação do usuário com o WfMS.

Camada de lógica de aplicação – É onde são executadas as funcionalidades para execução da definição do processo e da edição dos dados. Engloba os componentes de cliente do WfMS, e de servidor de banco de dados. Recebe solicitações feitas através da API, na primeira camada, e efetua a ligação com a terceira camada, onde está o servidor de banco de dados.

Camada de Servidores de Arquivos e Bancos de Dados – É onde estão as máquinas e o SGBD, que disponibilizam dados à segunda camada, executando as funções de gerência de base de dados.

2.8 APLICAÇÕES DE *WORKFLOW* E COMPONENTES

Aplicações baseadas em *workflow* descrevem um modelo de processo utilizando componentes de aplicação para a realização global das funções do fluxo. Devem ser adequadas a mudanças no processo, com um impacto mínimo sobre a implementação do mesmo. Um processo pode consumir alguns meses para sua conclusão, exigindo que a aplicação deva suspender a execução e que possa reiniciá-la posteriormente, recuperando o estado do processo. Aplicações de *workflow* precisam interagir com outros componentes de aplicações, e suportar mudanças globais do processo, sem afetá-lo. A necessidade de controle e auditoria, exige destas aplicações a habilidade de monitorar a execução, nos termos do modelo de processo original. Finalmente, uma aplicação de *workflow* precisa interagir com diversos usuários finais, disponibilizando os recursos necessários à realização do trabalho [Schm99].

A idéia básica da construção de aplicações baseadas em *workflow*, é a divisão da aplicação global em componente de controle de fluxo, e um grupo de componentes de aplicação, que executam os passos estabelecidos na definição do processo.

2.9 INVOCAÇÃO DE PROGRAMAS

Aplicações são invocadas pelo WfMS para a automatização de uma atividade ou para apoiar o usuário na execução de um item de trabalho. Os programas que são requeridos pelas atividades, precisam ser acessados pelo WfMS, levando em consideração o sistema operacional do ambiente e o método para invocação do programa que implementa a atividade, que pode ser síncrono ou assíncrono. Em invocação síncrona, o WfMS requer a execução do programa e espera o resultado. Em método assíncrono, o WfMS faz a requisição, continua a trabalhar e, quando o programa chamado termina sua execução, o resultado é repassado ao WfMS. Os depósitos de entrada e de saída são utilizados para a invocação de programas, disponibilizando os dados para execução e retornando os resultados. Deve ser realizado um processo de

mapeamento, para adequação das estruturas dos depósitos às das bases de dados manipuladas pelo programa invocado.

Segundo [Leym00], dentre os mecanismos de invocação, pode-se utilizar **Invocação de Métodos**, padrão para sistemas orientados a objetos, utilizando gerência de objetos e mediação de chamadas a métodos de objetos, em um ambiente distribuído. Isto é feito por um ORB – *Object Request Broker*, que libera o solicitador da mensagem da necessidade de saber a sua localização. Outro mecanismo de invocação é a **Chamada a Programas** (*Program Call*), fornecido pelo sistema operacional. Um mecanismo similar a este é o de *Chamada de Procedimentos Remotos*, ou RPC (*Remote Procedure Call*), que permite a invocação remota de um programa. O **Monitor de Processamento de Transações** (*TP Monitor*), é um mecanismo para aplicações que são executadas em computadores de grande porte. O WfMS, utilizando este mecanismo, deve prover o monitor TP com a entrada adequada, e deve interpretar a saída fornecida pelo mesmo. O único mecanismo assíncrono é o de **Invocação de Enfileiramento de Mensagens**, ou MQI (*Message Queueing Invocation*). Os sistemas de *workflow* devem fornecer condições para o usuário definir o mecanismo de invocação desejado, uma vez que isto envolve o mapeamento entre as estruturas de dados do WfMS, e as da aplicação.

2.10 CONTROLE DE ACESSO

O controle de acesso é necessário, para definir as permissões de cada ator envolvido em um sistema de *workflow*, monitorando e restringindo o acesso aos recursos do sistema, de forma a garantir a integridade das informações, bem como a sua confidencialidade. Diferentes métodos de controle de acesso vêm sendo utilizados, dentre os quais o paradigma **RBAC – Controle de Acesso Baseado em Papéis** (*Roles Based Access Control*) [Hags00]. Este método consiste na atribuição de direitos de acesso, não diretamente às pessoas, mas a papéis, de forma que, como os usuários podem desempenhar diferentes papéis, eles têm permissões correspondentes àqueles que correntemente desempenham.

O motor do WfMS deve permitir atribuição de funcionalidades, tal como a delegação por um usuário, de suas responsabilidades, de parte delas, ou de atividades

individuais a outro participante do *workflow*, buscando assegurar a continuidade da operação do sistema. De modo a garantir segurança, tal delegação de permissões de acesso deve ser assegurado que o usuário delegado herde as permissões necessárias de quem delega, apenas pelo tempo em que elas forem determinadas como válidas [Hags00].

2.11 HISTÓRICO

2.11.1 TECNOLOGIAS ANTECESSORAS A *WORKFLOW*

A tecnologia de sistemas de *workflow* não é nova, datando da década de 1970 os primeiros trabalhos em sistemas de automação de escritórios, voltados ao controle de procedimentos do trabalho. Em seguida, nos anos 80, o foco evoluiu para modelos de processo e, aos procedimentos de automação de escritório, foram adicionados os de digitalização de documentos, compondo WfMS que se mostraram limitados no atendimento das necessidades de seus usuários [Mano01].

Os primeiros sistemas de *workflow* foram desenvolvidos para atender a problemas específicos, contando com o auxílio de aplicações computacionais, especialmente adaptadas ao WfMS. Os novos sistemas integram aplicações, através de *interface*s API, que padronizam aplicações a serem invocadas pelo WfMS.

A seguir, uma breve consideração sobre cada uma das principais tecnologias que antecederam os sistemas de *workflow*.

Processamento de imagens e documentos – Documentos são componentes importantes em processos, e são transferidos entre os executores de atividades. O advento da digitalização de documentos exigiu dos sistemas, funcionalidades para a sua distribuição em meio eletrônico, bem como o controle do seu ciclo de vida [Holl95].

Correio eletrônico – A distribuição da informação, associada a regras de armazenamento e roteamento das mensagens, para participantes de processos, levou ao desenvolvimento de funcionalidades de *workflow* [Cruz98].

***Groupware* e Trabalho Cooperativo Baseado em Computador (CSCW)** – O trabalho sob a ótica de CSCW, está sustentado em três pontos, que representam os aspectos de Colaboração, Comunicação e Coordenação. Tecnologias que permitem a troca de informações, ou o compartilhamento de atividades entre um grupo de executores, pertencem à categoria de sistemas baseados em CSCW. *Group*ware baseia-se em trabalho em grupo, onde pessoas compartilham recursos na execução de um segmento de trabalho. A implantação de sistemas de *groupware* possibilitou a implementação de sistemas CSCW, trazendo automatização a processos de negócio. Isto foi possível, graças ao advento das redes locais e da plataforma de computação distribuída, cliente-servidor [Cruz98]. Sistemas de *workflow* estão mais fortemente ligados ao aspecto da coordenação de atividades, executadas para criar um trabalho final, cobrindo apenas um dos aspectos de *groupware* [Leym00].

Re-engenharia de Processos de Negócio (BPR) – O trabalho em *groupware* ocasionou a necessidade de realizar a reengenharia de processos, conduzindo à demanda por integração de ferramentas de análise, modelagem e definição de processos, e de análise dos efeitos sobre os papéis organizacionais, e as responsabilidades associadas aos mesmos [Holl95]. Atualmente, a tecnologia da informação (TI) desempenha papel fundamental no desenvolvimento de estratégias de negócio e na implementação de novos processos [Shar00]. Sistemas de BPR são utilizados para a modelagem de processos, etapa que antecede a implantação de um sistema de *workflow*.

Aplicações baseadas em transações – São as que implementam as funcionalidades relacionadas ao negócio de uma organização. A necessidade de controle mais eficiente de processos, demandou a integração de *workflow* a programas baseados em transações. Uma transação deve possuir atomicidade, ou seja, deve ser uma unidade atômica de processamento, executada em sua totalidade, ou não executada [Elma00].

2.12 TIPOS DE WfMS

*Workflow*s podem ser classificados segundo o seu valor para o negócio da empresa, aquele para o qual ela existe, e segundo sua repetição. Os processos que identificam a atividade da empresa, são os de mais alto valor de negócio. A repetição mede a freqüência com que um processo é executado, da mesma forma, e serve como

indicador da modelagem do processo. Segundo estas duas características, podem ser distinguidos quatro tipos de *workflow*: *ad hoc*, administrativo, colaborativo e de produção, ou orientado a transação.

Ad hoc – Com baixo valor de negócio e baixa taxa de repetição, permite liberdade na estruturação de processos e de atividades [Cruz00]. Este tipo de *workflow* permite liberdade na estruturação de processos e atividades, definidas e estruturadas de acordo com o documento processado no fluxo. Um exemplo de *workflow ad hoc* é o tipo orientado para *e-mail*.

Administrativo – Apresenta baixo valor de negócio e alta taxa de repetição. Trata o fluxo de documentos administrativos, com atividades de verificação e controles de correção de erros das informações [Schm99].

Colaborativo – Possui alto valor de negócio e baixa taxa de repetição, trabalhando com processos geralmente complexos, de grande importância para o sucesso da empresa [Leym00].

Produção – Apresenta alto valor de negócio e alta taxa de repetição, com grande volume de transações, acesso a grandes bases de dados, envolvendo regras que espelham muitas políticas de negócio e atividades multifuncionais, sendo feitas por vários departamentos [Cruz98]. Como implementa o negócio da empresa, envolve requisitos especiais de operação do sistema, e do ambiente em que está inserido, como a capacidade rastrear processos concluídos e em execução. Os requisitos do empreendimento para a execução em ambiente distribuído e heterogêneo, são o suporte a múltiplas plataformas, o gerenciamento do sistema, administração central, a obediência a padrões, e o sistema sofisticado de segurança. [Leym00]. Sistemas de *workflow* do tipo produção apresentam um cenário, onde diferenças de desempenho das atividades afetam processos, de maneira mais significativa que nos demais tipos.

Os WfMS de produção, disponíveis no mercado, são compostos por módulos de gerência do tráfego na execução do processo, de disponibilização de um ambiente para acesso a bases de dados, de comunicação com a rede mundial (Internet), de forma transparente ao usuário do WfMS, e de **interoperabilidade**, que é um requisito

fundamental, para que o sistema se encaixe em um padrão definido pela WfMC, abordado em detalhes no capítulo 3.

As referências ao desenvolvimento da proposta, neste trabalho, são feitas a **WfMS do tipo produção**.

Uma outra dimensão, para a classificação de WfMS, está baseada no grau de automação do *workflow*, definindo a independência da intervenção humana, na execução dos processos.

WfMS podem ser comparados a sistemas operacionais, pois estes proporcionam o ambiente para execução, gerenciando recursos computacionais, invocando programas, controlando a ordem de execução, e os dados comunicados entre os componentes. O WfMS considera as diferenças entre sistemas operacionais e utiliza o mecanismo de invocação do sistema operacional local, proporcionando APIs apropriadas para que os aplicativos necessários às atividades não tenham que lidar diretamente com o mesmo. Desta forma, os WfMS provêm às aplicações a mesma função que os sistemas operacionais disponibilizam a programas individuais [Leym00].

CAPÍTULO 3 - *WORKFLOW MANAGEMENT COALITION* (WfMC)

Neste capítulo são descritos os padrões de interoperabilidade para sistemas de *workflow*, definidos pela WfMC.

A modelagem, em sistema de *workflow*, utiliza técnica e linguagem específicas, que variam de acordo com o WfMS. Isto pode inviabilizar, entre outras, a comunicação entre diferentes implementações de modelos de processo e a migração entre sistemas, o que onera a evolução dos sistemas na organização, ou a torna estanque, na integração com outras empresas.

A WfMC – *Workflow Management Coalition*, é uma organização criada por empresas produtoras de sistemas de *workflow*, que desenvolve especificações para a implementação, garantindo interoperabilidade entre WfMS heterogêneos, e a integração de aplicações de *workflow*, com sistemas de informação existentes nas organizações. A WfMC trabalha em acordo com a OMG (*Object Management Group*) e a WARIA (*Workflow And Reengineering International Association*), compondo uma força de trabalho para garantir que WfMS possam agir em interoperabilidade, visando a proteção dos investimentos em sistemas de *workflow*.

A seguir, são descritos os principais pontos, que apoiam a padronização da WfMC, para a interoperabilidade entre WfMS.

3.1 COMPONENTES DE UM WfMS

A WfMC define dois grupos principais de funções para sistemas de *workflow*, resultando no meta modelo de funções executadas em tempo de construção, e no meta modelo de funções executadas em tempo de execução do *workflow*, conforme mostrado na Figura 3.1.

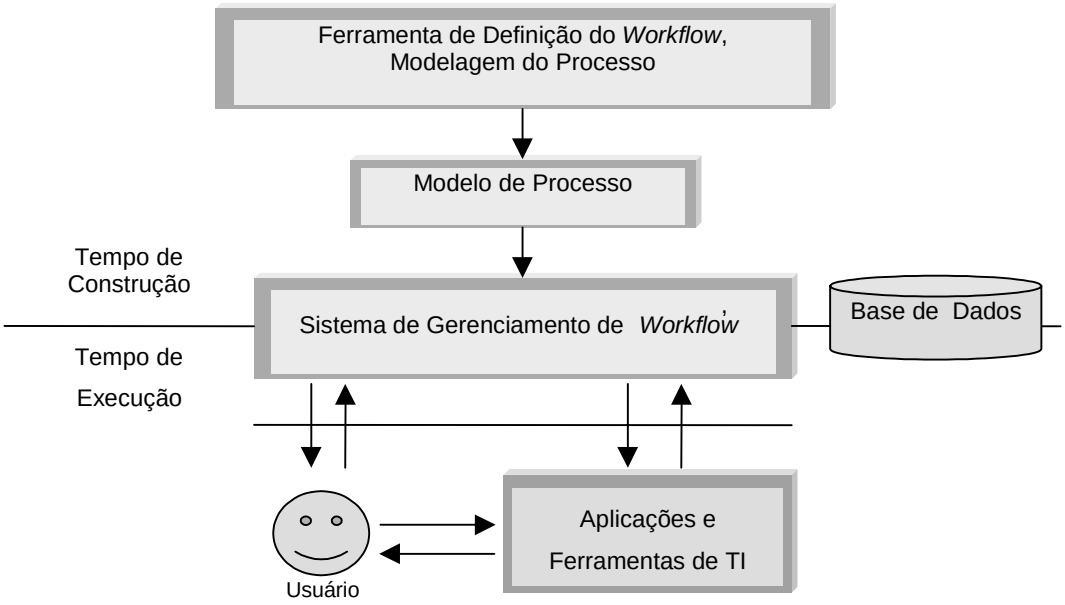

Figura 3.1 – Componentes de um WfMS [Leym00]

Funções em Tempo de Construção – Compreendem as funções executadas para definição do processo, das atividades, dos participantes do *workflow*, das condições de transição, dos dados de contexto e das aplicações que interagem com o WfMS. Estas funções podem ser desempenhadas, de acordo com a implementação do WfMS, através de uma *Interface* gráfica de usuário (GUI), ou de uma linguagem de definição de *workflow*.

Funções em Tempo de Execução – Resultam em uma definição computadorizada de um processo de negócios, que inclui as definições manuais, e de *workflow* [Holl95]. Estas funções têm como objetivo controlar a execução dos processos e das aplicações invocadas pelo WfMS. Elas identificam a definição do processo, uma instância do processo, uma instância da atividade e um item de trabalho.

Ferramenta de Definição *Workflow* e Modelagem do Processo – Cria a descrição do processo, em uma forma que possa ser processada por computador. É uma ferramenta que gera a *interface* para importar a definição do processo, e deve fornecer suporte a condições de início e término do processo, bem como a identificação das atividades dentro do mesmo, a identificação dos tipos de dados e caminhos, a definição das condições de transição e a informação para a alocação de recursos [Cruz98].

3.2 TERMINOLOGIA BÁSICA

A WfMC definiu termos básicos, presentes em sistemas de *workflow*, como forma de uniformizar as especificações na indústria de WfMS. A Figura 3.2, mostra os elementos mais importantes da especificação para WfMS,, definidos, em sua maioria, no capítulo anterior.

Figura 3.2 – Relacionamento Entre a Terminologia Principal da WfMC [Holl01]

3.3 PADRÕES PARA WfMS

Uma das tarefas principais dos WfMS é separar a lógica de processos, da lógica de atividades, embutidas em aplicações individuais de usuários. Esta separação permite que ambos sejam modificados independentemente, e que a mesma lógica de atividade possa ser reutilizada em diferentes processos, integrando aplicações isoladas e heterogêneas. Isto exige funcionalidades gráficas, que facilitam a documentação do processo, gerando arquiteturas pesadas, difíceis de reutilizar e de adaptar, demandando o uso de grande número de *framework*s, ferramentas de desenvolvimento e aplicações.

Transações de negócios, envolvendo companhias ao redor do mundo, exigem a interoperabilidade entre WfMS de diferentes fabricantes. Facilidades para integração com aplicações de negócio, são pontos cruciais para que seja viável a implantação de um sistema, pois permitem que processos possam ser executados por equipes, ou indivíduos, distribuídos geograficamente, operando vários WfMS, definindo *subflows*, ou *workflows* que complementam a ação de um *workflow* maior, interagindo através da troca assíncrona de mensagens. Como forma de padronizar a definição de mensagens, a linguagem XML têm sido utilizada. A diversidade dos WfMS vai, desde motores de *workflow* específicos, a aplicações e funcionalidades de *workflow*, implementadas em servidores de objetos de negócio [Schm99].

A WfMC adotou como estratégia para a implantação de um padrão para WfMS, o estabelecimento de um vocabulário para *workflow*, seguido da publicação da especificação da API para esses sistemas (WAPI), que cobre a *interface* para aplicações-cliente, e a especificação de interoperabilidade para *workflow*. A esta *interface*, se seguiram padrões para dados de auditoria, importação e exportação de definições de processo, modelo comum de objetos, e uma especificação baseada na linguagem XML [Fann01].

Atualmente, a padronização ocupa espaço na definição de objetos de negócio e ambientes de roteamento de mensagens (*message-brokering*). Quando o WfMS age como *message broker*, ele permite o processamento de requisições complexas, que exigem a coordenação de diversas aplicações e controle sobre os resultados, além de constituir uma ponte de ligação com os sistemas legados da organização.

Há implementações que envolvem a combinação de componentes de negócio com *message broker*, sistemas ERP e *subflows*, na integração de processos complexos. Neste caso, o uso de objetos de negócio promove uma visão natural para descrever entidades, pois a tecnologia de *workflow* permite a construção de aplicações orientadas a atividades, de modo a isolar o desenvolvimento de funções de negócio, de sua disponibilização em um ambiente de execução, possibilitando uma visão de *software* livre de conceitos, como base de dados, ferramentas e aplicações [Schm99]. A aplicação desenvolvida por profissionais de TI, apenas irá prover o ambiente para execução destes objetos [Orfa97].

A definição de padrões, destinados a especificar APIs e protocolos, para a interação de motores de *workflow*, evoluiu para a especificação de meta modelos e *interface*s abstratas, que podem ser implementadas em diferentes áreas de aplicação e ambientes de execução. O padrão para a facilidade de gerenciamento de *workflow* (*Workflow Management Facility*), da OMG, adapta os padrões da WfMC para uma infra-estrutura de objetos de negócio.

3.4 META MODELO DE WfMS

Um meta modelo de WfMS é uma estrutura, na qual podem ser construídos modelos de processo. O meta modelo, estabelecido pela WfMC para definições de processo, é mostrado na Figura 3.3.

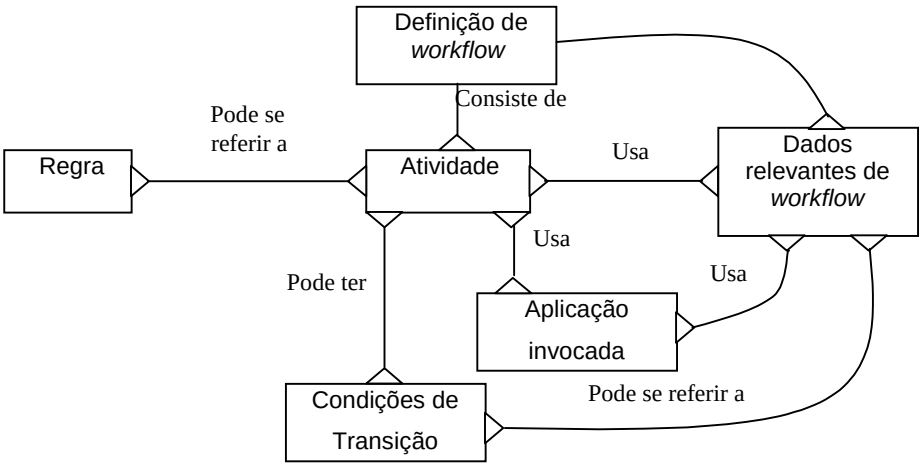

Figura 3.3 – Meta Modelo Básico de Definição do Processo [Holl95]

O modelo engloba a estrutura organizacional do ambiente onde o WfMS está inserido, o modelo de processos, a estrutura das listas de trabalho para os participantes do fluxo, os programas a serem invocados para execução das atividades, o ambiente operacional que suporta o sistema, entre outros. Esta definição é feita, considerando funções executadas em tempo de construção do modelo, funções em tempo de execução, e a base de dados com as informações gerenciadas pelos componentes de tempo de construção e de execução. O meta modelo da WfMC define um grupo básico

de propriedades destes elementos, e provê mecanismos de extensibilidade, para adicionar novas propriedades [Schm99].

A estrutura organizacional do ambiente do WfMS deve permitir que os participantes humanos sejam identificados pelos papéis funcionais que desempenham, tornando o modelo independente da rotatividade de indivíduos na organização.

3.5 MODELO DE REFERÊNCIA PARA WfMS

O modelo de referência de *workflow*, da WfMC, foi criado em outubro de 1993, definindo um padrão a ser adotado por fabricantes de WfMS, que exige que um *software* de *workflow* possa se conectar a diversos tipos de SGBD, através de múltiplos padrões de redes, com equipamentos de diversos fabricantes [Holl95].

O modelo de referência introduz a abstração de um serviço que represente qualquer tipo de ambiente de execução de *workflow*, para um componente de controle de fluxo de aplicação de *workflow*. O componente de controle de fluxo pode ser implementado por um objeto de negócio, residindo em um servidor de objetos de negócio, ou por um modelo de processo, representado por um motor de *workflow*, cujos detalhes internos estão fora do controle de padronização [Schm99].

As áreas de interações entre WfMS e componentes desenvolvidos independentemente, incluem o intercâmbio de definição de processo, o comportamento esperado de um processo, em qualquer ferramenta de modelagem, e a realização deste comportamento em qualquer infra-estrutura de execução de *workflow*, além das interações entre aplicações de *workflow* e os participantes, para controlar a execução do processo e manipular itens de trabalho, de uma forma unificada. O modelo de referência ainda engloba *interfaces*, para a interoperabilidade entre aplicações de *workflow*, permitindo que aplicações desenvolvidas independentemente, e gerenciadas por *workflow*, interajam na execução de processos complexos [Alle01]. A Figura 3.4 detalha as cinco áreas de funcionalidades entre um WfMS e o ambiente de seu contexto, cobertas pelo modelo.

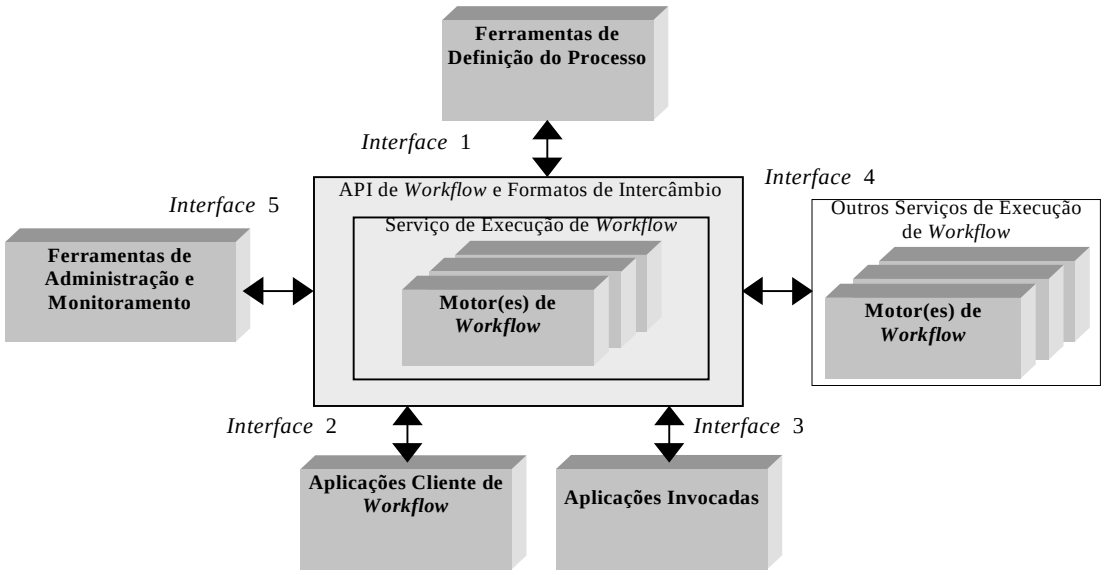

Figura 3.4 – Modelo de Referência de *Workflow* da WfMC [Holl01]

A seguir, são comentadas as *interface*s do modelo

***Interface* 1 – Definição do Processo** – Cuida dos padrões para definição de processos, de forma que diferentes ferramentas possam ser utilizadas na preparação de processos a serem executados. Nesta *interface* foi definida a linguagem PDL, *Process Definition Language*, de definição de processos [Schm99].

Interface*s 2 e 3 – API de Programas de Aplicações de *Workflow – Cobrem os aspectos relativos às *interfaces* de programas de aplicação em *workflow* (WAPI), e regulamentam a integração de ferramentas de terceiros, que necessitam acesso ao motor do *workflow*, tais como agenda e correio eletrônico. O componente manipulador de listas de trabalho deve obedecer aos padrões desta *interface* [Alle01]. A API de controle de processos de *workflow*, provê operações para criação de instâncias de processos, o início e fim da execução destas instâncias, e a obtenção dos resultados [Schm99].

Interface* 4 – Inter-Motores de *Workflow – Relaciona os aspectos relativos às chamadas que um sistema de *workflow* faz a outros sistemas, envolvendo cooperação entre seus motores, para cumprir com as atividades de processos distribuídos entre organizações. A interoperabilidade a ser atingida, coberta por esta *interface*, especifica que em um ambiente com vários WfMS, uma organização possa produzir uma definição

de processo, e esta deva apresentar o mesmo comportamento, independentemente de qual motor de sistema de *workflow* o execute [Alle01].

***Interface* 5 – Monitoramento e Auditoria** – A função desta especificação é padronizar dados consistentes, para análise de execuções em WfMS heterogêneos, que servem a propósitos de auditoria e de análise de processos executados. Informações de auditoria permitem que se determine dados de histórico de atividades, a partir dos quais os parâmetros para distribuição de atividades, entre os participantes do *workflow*, podem ser definidos [Alle01]. Esta *interface* determina o padrão da WfMC para especificação de quais informações precisam ser armazenadas, a partir dos vários eventos de mudança de estado, que ocorrem durante a execução do *workflow* [Schm99].

3.6 MODELO ABSTRATO DA ARQUITETURA DE UM WfMS

A WfMC define um modelo abstrato de arquitetura *workflow*, recursivo o suficiente para abranger a integração de *workflow* com outras tecnologias. A Figura 3.5 mostra os componentes deste modelo de implementação.

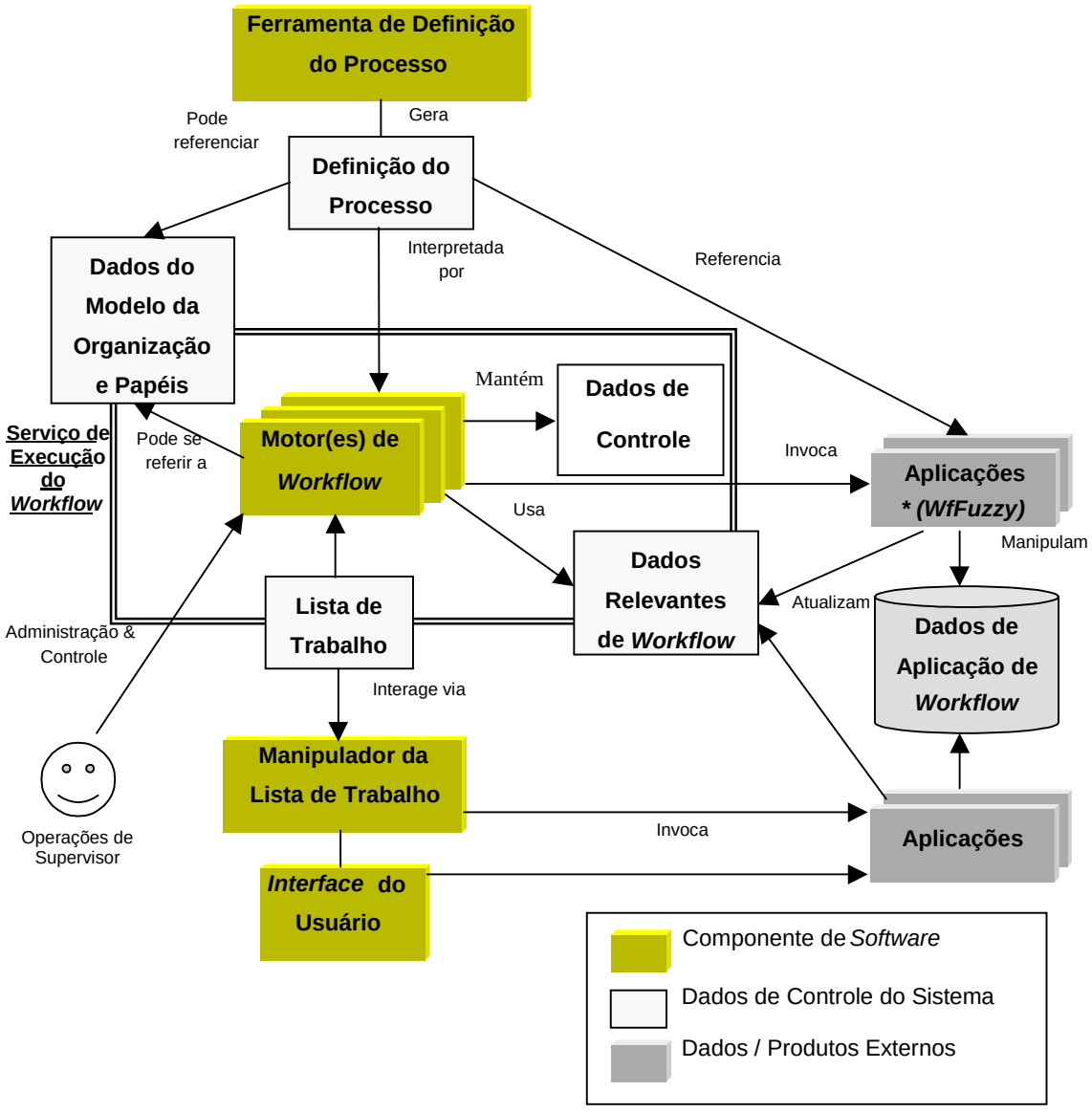

Figura 3.5 – Modelo Abstrato de WfMS da WfMC [Holl95]

A seguir, são fornecidos detalhes dos elementos desta arquitetura, ainda não citados no capítulo 2 deste trabalho. As citações são feitas pela WfMC [Holl01], à exceção das que forem referenciadas como de outros autores.

Lista de Trabalho – A lista agrupa itens de trabalho, segundo características comuns, devendo permitir a quem modela um processo, definir sua forma e conteúdo. Uma lista pode ser associada a um ator, a um processo específico, a uma atividade, ou pode se referir a um papel funcional na organização. Neste caso, todos os atores que tiverem o papel definido na lista, podem ter acesso aos itens nela contidos. O item pode ser retirado da lista de trabalho, quando o ator iniciar sua execução, ou continuar na lista, mas sendo apresentado de uma forma relativa ao seu novo estado [Leym00].

Manipulador da Lista de Trabalho – É um componente de *software* que gerencia a interação entre o usuário e a lista de trabalho, mantida por um motor de *workflow*, requisitando deste, durante a criação da lista de trabalho, os itens associados à instância da atividade, estabelecidos na definição do processo. Ele permite que itens de trabalho sejam passados do WfMS para os usuários, e que as notificações de término, ou outras condições de estado do item de trabalho, sejam passadas entre o usuário e o WfMS. O manipulador da lista também pode sintetizar em uma única lista, itens de trabalho de vários WfMS, além de realizar a invocação de ferramentas, ou de aplicações, como parte do processamento de itens de trabalho [Holl01].

Serviço de Execução do *Workflow* – Interpreta a descrição e controla a instanciação do processo e suas atividades, através da ação de um ou mais motores de *workflow* mantendo, entre outras, informações de estado de processos e de atividades em execução, centralizadas ou distribuídas pelo grupo de motores. A definição do processo é usada em conjunto com os dados relevantes de *workflow* em execução, para controlar a navegação, através dos passos das atividades [Holl95].

Dados de Aplicação – Dados específicos, gerenciados pelas aplicações, em interações com usuários, em tempo de execução de processos. Dados da aplicação compreendem os dados gerados e utilizados por cada atividade, ao longo do *workflow*, e não são acessíveis através do WfMS [Silv01].

Dados Relevantes de *Workflow* – Dados usados pelo WfMS, para determinar as transições de estado de uma instância de *workflow*, como pré e pós condições, condições de transição, ou assinalamento de participantes de *workflow*. Estão na fronteira entre os dados de aplicação e os de controle, pois podem ser utilizados, tanto por aplicações do cliente, como pelo motor do WfMS. "Dados relevantes compreendem a definição do fluxo de trabalho, com suas atividades, informações de estado e de histórico, de cada instância do *workflow*, e qualquer outro tipo de dado relativo à execução do *workflow*" [Silv01]. A associação de uma atividade a um executor ocorre na geração dos dados relevantes de *workflow*, referenciando a execução, ou destruição das instâncias das atividades. A partir destes dados, os motores de *workflow* podem criar ou destruir entradas de referências às instâncias, na lista de trabalho.

Dados de Controle – São dados internos ao WfMS, gerenciados pelo motor de *workflow*, não acessíveis a aplicações. Estes dados representam o estado dinâmico do WfMS e suas instâncias de processo.

Aplicação de *Workflow* – Termo genérico para um programa de *software* que interage com o serviço de execução do *workflow*, assumindo parte do processamento requerido, para auxiliar atividades.

***Interface* do Usuário** – Controla a *interface* local com o usuário, podendo ser combinada com o manipulador de listas de trabalho, em uma única entidade [Holl95]. O fato do motor do *workflow* fazer uso da lista de trabalho, permite que as alterações realizadas por usuários, via *interface* do WfMS, e feitas por agentes de *software* representando aplicações, possam conviver entre si. Esta *interface* determina o poder de processamento de um *software* de *workflow*, que é tanto maior, quanto maior for a sua possibilidade de compartilhar dados entre seus próprios módulos e programas de outros fabricantes [Cruz98].

***Interface* Gráfica do Usuário (GUI)** – É um componente que torna os itens de trabalho, as listas e os processos, disponíveis para visualização, de forma que os usuários possam decidir que item executar, acompanhar o estado de uma instância de processo, ou de atividade, entre outros. Itens, listas e processos, podem ser

disponibilizados pelo WfMS como ícones, pastas, ou documentos, obedecendo ao padrão de exibição de arquivos do sistema operacional.

Operações de Supervisor – Operações de supervisor permitem que o administrador do *workflow* intervenha em regras de alocação, faça o rastreamento do histórico de uma instância de processo, a fim de obter informações estatísticas, entre outras [Holl95].

3.7 SEGURANÇA

Aspectos de segurança, envolvendo sistemas de *workflow* compreendem, além de fatores ligados à integridade, requisitos de identificação, necessários para a comunicação entre atores, em um fluxo, ou entre diferentes fluxos de trabalho [Holl95].

Quando a comunicação envolve interoperabilidade entre dois ou mais WfMS, um sistema deve saber de qual par determinada mensagem vem, e para qual ela vai, com segurança. Aspectos de segurança envolvem confidencialidade, integridade, disponibilidade, autenticação, autorização, auditoria, anonimato e separação de deveres [Atlu01]. Destes, vale ressaltar a **autenticação**, definida pela WfMC como o processo de identificação, para o sistema de computadores, das diversas funções que os usuários podem executar, baseadas no papéis estabelecidos na definição do processo [Holl95].

CAPÍTULO 4 - INTELIGÊNCIA ARTIFICIAL

Neste capítulo são apresentados alguns conceitos relativos inteligência artificial (IA) e são abordados dois paradigmas de IA, que podem ser utilizados na implementação da proposta de distribuição inteligente de atividades entre atores humanos, em sistemas de *workflow*. Os aspectos abordados visam fornecer um embasamento para justificar a escolha do paradigma adotado no trabalho.

O conceito de inteligência artificial (IA), permite uma diversidade de opiniões, sobre o que na realidade pode ser considerado como inteligência, quais seus limites e potencialidades, sua modelagem em computadores, e quão inteligentes os sistemas podem ser. Uma definição nos permite compreender a natureza do assunto, em um nível adequado para este trabalho. De acordo com [Barr00], citando Eugene Cherniak e Drew McDermott, a definição de Inteligência Artificial como o estudo das faculdades mentais, com o uso de modelos computacionais, é uma das que mais encerra características de artificial, cobrindo de maneira mais ampla sua aplicabilidade a sistemas de computadores, e sua relação com ciências afins.

Formas de implementação de IA fazem uso de inteligência computacional, que consiste em um grupo de técnicas envolvendo computação, que exibem uma habilidade de aprendizado e/ou manipulação de novas situações, de modo que o sistema funcione com atributos de racionalização [Eber96].

Comportamento inteligente pode ser produzido pela manipulação de símbolos, tratados em computador, através da representação por cadeias de caracteres, ou números, processados por algoritmos, especialmente desenvolvidos para estes símbolos [Bigu98].

Sistemas implementados com princípios de inteligência artificial permitem que, em um computador, seja refletido o comportamento de adaptação a situações, que mudam em um determinado contexto, como acontece com os seres humanos. A ocorrência repetida de uma determinada situação, permite ao sistema captar o

conhecimento a respeito do comportamento. Quando este padrão de comportamento for alterado, o sistema estará apto a identificar a variação e poderá gerenciar o problema.

Técnicas como mineração de bases de dados (*data mining*), controle adaptativo e componentes de *software* atuando em redes, de forma autônoma, inteligente e com mobilidade, têm feito uso de redes neurais, algoritmos genéticos e de lógica nebulosa [Bigu98].

Problemas envolvendo decisões, podem ser classificados como de decisão simples, ou múltipla [Barr00]. O caso da escolha do roteamento do fluxo de trabalho em sistemas de *workflow*, objeto deste trabalho, pode ser considerado de decisão simples, pois não envolve várias ações a serem tomadas, mas diversas variáveis para a tomada de uma única decisão.

A seguir, serão comentados os aspectos dos paradigmas de Redes Neurais e Lógica Nebulosa, considerados para aplicação no trabalho.

4.1 REDES NEURAIS ARTIFICIAIS (RNA)

As redes neurais artificiais simulam o comportamento do cérebro humano, com uma estrutura de neurônios artificiais, dispostos em camadas, determinando sua topologia. Outros elementos básicos da RNA são as conexões entre os neurônios e as regras que determinam o aprendizado da rede.

Alguns conceitos importantes para o entendimento deste tópico, são fornecidos a seguir:

> "Uma RNA é um paradigma de análise, modelado de acordo com um paralelismo com a estrutura da mente." [Eber96].

> "Uma RNA apresenta comportamento emergente, em resposta aos estímulos do ambiente" [Bigu98].

Em uma RNA, os **neurônios** são chamados de **elementos de processamento** (EP) que, interconectados, são responsáveis pela não linearidade da rede. Os EP são

inspirados no modelo nervoso da mente, e no processo de neurotransmissão, neste caso, feito através do processamento interno de certas funções matemáticas. O primeiro passo na especificação de uma RNA é a definição do número de EP, e sua disposição em camadas, de entrada, intermediárias e de saída. A camada de entrada, recebe informações do ambiente e distribui para uma ou mais camadas, em um nível intermediário, onde funções de combinação e ativação são responsáveis pelo processamento da informação recebida. No terceiro nível, está a camada de saída, ou de resposta da rede, responsável pela distribuição da informação processada no nível anterior [Galv99].

Os EP são compostos por **pesos sinápticos**, que ponderam os valores das entradas, pela **regra de propagação**, que define como as entradas são combinadas no EP, de acordo com seus pesos, e a pela **função de ativação**, que determina o efeito que a regra de propagação terá sobre o nível de ativação do EP. A **função de saída do EP** define uma função de transferência, que têm por finalidade relacionar o estado de ativação conhecido de um EP, com o seu sinal de saída.

Os **pesos sinápticos** definem o conhecimento do sistema e a importância de cada informação, com relação a um dado neurônio. A representação de um problema é tanto melhor, quanto melhor for a definição do conjunto de pesos.

O **aprendizado** de uma rede neural envolve a modificação de seu padrão de interconexão, podendo ser feito com o estabelecimento de novas conexões, a eliminação de conexões existentes e a alteração dos pesos das conexões.

O aprendizado é **supervisionado**, quando a rede é treinada com vetores de exemplos compostos de entradas e saídas. Quando o erro entre os padrões de saída atingem um valor mínimo desejado, a rede está treinada para sua tarefa. Desta forma, ela adquiriu o conhecimento que a pessoa, responsável pelo seu ajuste, tinha do ambiente. Redes do tipo *Perceptron* multicamadas, utilizam este tipo de aprendizado.

O aprendizado **não supervisionado**, ocorre quando a rede aprende com os próprios dados de entrada, ou seja, o algoritmo de treinamento não requer o conhecimento prévio das saídas. À rede são dadas condições para realizar uma medida e

os parâmetros livres da rede são otimizados, de acordo com um algoritmo, em relação a essa medida [Hayk01].

O tempo de treinamento têm relação direta com o tamanho da rede, definido pelo número de EP nas camadas. O treinamento nem sempre envolve uma função simples, de forma que a solução encontrada pode apresentar mínimos locais que não correspondam à solução ótima.

A verificação da rede é feita utilizando-se um conjunto de dados que não foi empregado durante o treinamento. Caso o resultado não seja satisfatório, uma nova topologia para a rede é escolhida, e o ciclo é repetido. A rede estará pronta para ser utilizada, depois de feito o treinamento e sua verificação.

Utilizando um algoritmo de aprendizado com propagação recursiva de erros (*backpropagation*), o usuário estabelece um limite aceitável de erro, resultado da comparação entre a saída desejada, e a saída real da rede, e a rede executa ciclos de treinamento, estabelecendo pesos aleatórios para começar o trabalho. Em cada ciclo, a rede calcula uma saída, o erro quadrado, atualiza os pesos e calcula um erro cumulativo do ciclo, a ser comparado com o limite de erro aceitável. O algoritmo é encerrado quando é obtida uma solução aceitável, ou quando é executado um limite de ciclos pré estabelecido [Klir95].

As redes neurais tem aplicação mais importante em reconhecimento de padrões, como o reconhecimento da fala, de formas físicas ou da escrita.

As redes neurais apresentam resposta imediata e, uma vez treinadas, consomem um mínimo de ciclos de processamento do computador para a sua execução [Bigu98]. Outra característica importante de uma rede neural é a sua capacidade de generalização, que permite que ela se adapte a uma mudança na entrada, baseada no conhecimento adquirido do problema.

Para aplicação neste trabalho, uma rede neural com aprendizado supervisionado, atuaria sobre dados padrão de comportamento dos atores, para os diversos tipos de atividades, em processos de negócios envolvendo WfMS. Todavia, como o elemento humano que gerencia os sistemas de *workflow* tem conhecimento voltado a atividades

administrativas e de negócio, a dificuldade de modelar os dados para o treinamento, e de definir a melhor topologia da rede, limita sua aplicação a um aperfeiçoamento futuro do trabalho, proposto no capítulo 6.

4.2 LÓGICA NEBULOSA

A lógica nebulosa foi desenvolvida para suprir a representação de apenas dois estados da lógica binária, ou booleana. Em lógica nebulosa, o domínio de uma variável é chamado de **universo de discurso**, dividido em regiões lingüísticas, chamadas de **conjuntos nebulosos**, ou faixas de valores definidos entre os números reais. Quanto maior a distância entre os limites máximo e mínimo do conjunto, mais impreciso, ou nebuloso, é este valor. A intensidade com que um valor discreto ***x***, no universo de discurso, pertence a um conjunto ***A***, é ditada pelo grau de pertinência ***m***, no conjunto, determinado por uma função do tipo $\mu A(x)=m$ [Bigu98], [Klir95].

A racionalização com lógica nebulosa, aborda o grau de intensidade da ocorrência de um evento.

Sistemas de lógica nebulosa permitem a representação do conhecimento por expressões lingüísticas, não precisas, familiares aos seres humanos. O ser humano traduz valores numéricos exatos, em lingüísticos e vice-versa, baseado em uma escala pessoal de valores, e utiliza esta capacidade para tomar decisões. A lógica nebulosa apresenta a vantagem de facilitar a tradução do conhecimento. Em sistemas administrativos, certo grau de imprecisão é tolerável e irredutível. O pensamento humano compensa esta imprecisão codificando e rotulando a informação relevante para suas tarefas, em conjuntos nebulosos que guardam relação com os dados primários [Zade73]. Um sistema de lógica nebulosa implementa valores lingüísticos com um mecanismo de controle estável, pouco afeito às variações advindas de mudanças no comportamento humano.

A incerteza em sistemas nebulosos pode ser resultante da falta de informação, ou da necessidade de considerar apenas determinado grau de certeza, através de uma quantificação, que reduza a informação relativa à variável, a um nível necessário para

uma tarefa. Desta forma, é reduzida a complexidade do uso do sistema para um fim específico [Klir95].

Controladores nebulosos permitem que se reduza o grau de incerteza, existente em sistemas baseados em lógica binária, a patamares desejáveis para a realização de uma tarefa. Um controlador nebuloso é composto de uma base de regras nebulosas, um motor de inferência nebulosa, e dos módulos de fuzificação e defuzificação.

4.2.1 FUZIFICAÇÃO

Fuzificação é o processo de aplicação de valores pontuais aos conjuntos nebulosos, e a determinação de uma função de pertinência do valor a cada conjunto, feita por um difusor. O difusor produz conjuntos nebulosos, que são combinados para gerar um conjunto nebuloso simples, que é então defuzificado. Os tipos de difusores determinam o tipo de conjunto gerado. Difusores de ponto único (*singleton*) geram conjuntos com função de pertinência pontual.

A atividade de processamento no processo de fuzificação pode ser reduzida, se forem definidos pontos de avaliação de pertinência para a variável. A definição do número de pontos necessários para manter a precisão da fuzificação de uma variável pode ser obtida a partir do conhecimento desta variável, no domínio da aplicação do sistema, mas a partir de 150 pontos o erro de fuzificação cai para menos de 1% [Shaw99].

4.2.2 DEFUZIFICAÇÃO

O processo de defuzificação, feito a partir da aplicação de um difusor ao conjunto resultante da fuzificação, gera um valor pontual que representa a distribuição das pertinências nos conjuntos da variável de saída [Bigu98]. O propósito da defuzificação é gerar valores que representam ações tomadas pelo controlador nebuloso nos ciclos de processamento [Klir95].

Os métodos de defuzificação mais utilizados são: Centro-de-Area (C-o-A), Centro-de-Máximo (C-o-M) e Média-de-Máximos (M-o-M), apresentados a seguir.

Defuzificação por Centro-da-Área (*Center-of-Area* – C-o-A)

Neste método é determinado o centro de gravidade da figura, ou centróide, resultante da união dos conjuntos nebulosos, envolvidos na avaliação das regras de inferência, sobre a variável de saída [Klir95].

Defuzificação por Centro-de-Máximo (*Center-of-Maxima* – C-o-M)

Este método determina os valores máximos de pertinência, considerando a altura resultante da projeção do valor discreto, no conjunto onde ele ocorre. O valor defuzificado é um ponto de equilíbrio, calculado como uma média dos pesos do termo de máxima pertinência do vetor de possibilidades, ou pertinência *singleton*. As áreas das funções de pertinência não são consideradas, mas sim as múltiplas contribuições das regras ativadas [Shaw99].

Defuzificação por Média-de-Máximos (*Mean-of-Maxima* – M-o-M)

Neste método, o valor resultante é determinado a partir da média dos valores discretos no universo de discurso, decorrentes dos valores nas funções de pertinência onde a entrada discreta ocorre [Klir95]. O método M-o-M é utilizado em casos onde a função de pertinência tenha mais de um valor máximo e desconsidera o formato da função de pertinência de saída. Este método é descontínuo, ou seja, pode apresentar mais de uma solução, e um melhor compromisso entre possíveis saídas, com multiplicidade de disparo de conjuntos nebulosos, pode saltar para um valor diferente, com apenas uma pequena mudança nas entradas. Este método é adequado para o reconhecimento de padrões [Shaw99].

Os métodos **C-o-A** e **C-o-M** são contínuos, e o resultado na saída não varia para um valor diferente, com pequenas alterações na entrada, o que é indicado para o problema proposto, uma vez que os dados tratados são de natureza quantitativa. Ambos apresentam resultados similares, para a forma triangular e trapezoidal das funções de pertinência utilizadas. Há uma pequena diferença na precisão, a favor do cálculo do centro da área, todavia, o cálculo requerido para a determinação da área da figura resultante do conjunto nebuloso, demanda consideravelmente mais recursos

computacionais do que a determinação dos pontos máximos de pertinência, no método C-o-M.

Como a aplicação proposta envolve a utilização simultânea do mecanismo de inferência, por grande número de usuários, o método **C-o-M** é preferível, neste caso, por trazer ganho de desempenho ao sistema como um todo.

A operação de um controlador nebuloso ocorre em ciclos de quatro passos [Klir95]:

- Medição das variáveis relevantes do processo controlado;
- Conversão das medidas em conjuntos nebulosos que expressem a incerteza da medição, este passo sendo chamado de fuzificação;
- Avaliação, a partir dos valores fuzzyficados, pelo motor de inferência, das regras armazenadas na base de regras nebulosas. Isto gera um ou mais conjuntos, definidos no universo de ações possíveis;
- Conversão do conjunto resultante em um valor, ou vetor de valores discretos, ou defuzificados, que, em certo grau, melhor representam o conjunto nebuloso. Os valores resultantes do método de defuzificação representam ações tomadas pelo controlador nebuloso.

Uma questão que deve ser levada em consideração é a determinação dos universos de discurso e de seus conjuntos nebulosos. Os conjuntos nebulosos podem ser definidos com funções de pertinência trapezoidais, triangulares, gaussiana, sigmoide, ou *spline* cúbico (*S-shape*) [Shaw99]. Conjuntos triangulares, mais simples, apresentam uma pequena alteração de desempenho, em relação ao grande aumento computacional demandado para o cálculo de uma função curvilínea, como a gaussiana, a sigmóide ou *S-shape*, mais complexas [Eber96]. Além disso, o conhecimento matemático requerido para modelar, pouco popular, pode prejudicar a definição de funções de pertinência curvilíneas. A imprecisão de certos sistemas faz com que o modelo matemático dos mesmos seja de difícil implementação e possa ser não representativo da realidade. Modelos matemáticos exigem conhecimento pouco disponível nas organizações.

Modelos baseados em método heurístico, por sua vez, envolvem a experiência das pessoas na definição das regras de inferência, relacionando antecedentes e conseqüentes.

Na inferência nebulosa, a relação entre antecedentes e conseqüente determina uma função de entrada e saída, que têm especificados conjuntos de valores possíveis de pontos na função.

A precisão das funções de pertinência é afetada pela concentração próxima do ponto de equilíbrio que em funções triangulares é densa, garantindo maior sensibilidade para um ajuste de posição, e pela sobreposição das funções de pertinência, considerada adequada, em 50% [Shaw99].

Especialmente em sistemas de lógica nebulosa, padrões de raciocínio são a base para a definição de parâmetros.

Sistemas nebulosos têm a vantagem de serem mais facilmente assimilados, na definição de variáveis e seus intervalos, mas pequenas alterações nos conjuntos nebulosos podem alterar completamente o resultado esperado.

A definição das funções de pertinência, pode ser feita a partir das amostras de dados reais disponíveis, baseada na teoria matemática da adequação de curva, utilizando **Interpolação Lagrange**, que realiza uma adequação perfeita entre a função de pertinência e os dados utilizados, mas têm complexidade crescente de acordo com o volume de amostras, e exige que a função resultante seja criticamente avaliada e corrigida, ou através do **Ajuste de Curva por Mínimos Quadráticos**, que requer a escolha de uma classe adequada de funções parametrizadas, permitindo a determinação de funções, baseada em experiência anterior, ou a comparação entre classes de funções [Klir95]. Os dois métodos, todavia, envolvem alto conhecimento específico da teoria de sistemas nebulosos. No trabalho, a definição dos pontos das funções foi realizada através do conhecimento prático obtido com as pessoas das organizações citadas na página 75. Desta forma, buscou-se comprovar a facilidade de modelar a solução a partir do conhecimento das pessoas envolvidas nos processos de negócio.

4.3 CONSIDERAÇÕES E SOLUÇÃO ADOTADA

Em processos de negócio, a modelagem das variáveis que determinam a execução de atividades exerce papel importante como mecanismo de controle. Nestes casos, a formulação de um modelo matemático para o processo envolve uma dificuldade, acrescida pela necessidade de representação do comportamento humano. A definição de um modelo baseado em experiência pessoal, é facilitada pela possibilidade de agregar o conhecimento de qualquer pessoa importante para o processo, não obstante o seu grau de instrução, ou de especialização técnica, mas com base em seu conhecimento heurístico. Isto constitui uma vantagem a favor de sistemas de lógica nebulosa, sob o ponto de vista da maior disponibilidade de especialização exigida, e do comprometimento do usuário final com o sucesso da implantação.

O desenvolvimento da proposta, baseado em lógica nebulosa, se deve à configuração simplificada dos parâmetros para inferência, em relação à adequação da topologia e do treinamento que seria exigido para uma rede neural.

A aplicação proposta, detalhada no capítulo 5, permite definir os conjuntos difusos de forma interativa, determinando que o conhecimento heurístico, exigido para representar o comportamento esperado dos atores em um WfMS, possa ser facilmente registrado no sistema, para a tomada inteligente de decisão.

No trabalho serão utilizados o método **C-o-M** e conjuntos com forma **triangular** e **trapezoidal**, pelas razões explicadas na seção anterior.

CAPÍTULO 5 - DESENVOLVENDO O WfFuzzy

Neste capítulo é descrita a abordagem adotada para o desenvolvimento da proposta, e são detalhados os elementos de interação do usuário com o sistema. Também fazem parte do capítulo a descrição das variáveis nebulosas, consideradas para a avaliação de desempenho dos atores, e sua adequação para a execução de uma atividade, em um sistema de *workflow*. Para a comprovação da funcionalidade de distribuição de atividades, é necessário utilizar o sistema desenvolvido, invocado por um WfMS. Como o objetivo deste trabalho não é o desenvolvimento de um sistema de *workflow*, e por não estar sendo utilizado um sistema existente no mercado, foi desenvolvido um protótipo chamado WfFuzzy, com funcionalidades simplificadas de WfMS, recursivas o suficiente para comprovar a distribuição de atividades aos atores. O protótipo espelha as características mencionadas no capítulo 3, na seção 3.6, do presente trabalho.

Os recursos do WfFuzzy envolvem a definição simplificada do processo, a exibição da atividade a ser executada e a sua distribuição, sendo exibida na lista de trabalho do ator escolhido. A execução do item de trabalho, consiste em assinalar a data e a hora de início e fim da execução. Como forma de registrar um controle de qualidade sobre o trabalho executado, é registrada a aceitação ou a rejeição da execução, pelo administrador do *workflow*. Os demais aspectos são abstrações de um sistema real, enquadradas no padrão da WfMC. Esta limitação se deve ao fato de que o tema central do trabalho é a escolha do executor da atividade, através de inteligência artificial. Com isto, o trabalho está considerando a aplicação da solução proposta a um WfMS, em aspectos de interação do sistema, para a distribuição de atividades.

5.1 METODOLOGIA

A abordagem para o desenvolvimento do trabalho partiu da identificação de um problema, relativo a desempenho em sistemas de *workflow*, e da formulação de uma proposta de solução, utilizando recursos computacionais.

5.1.1 MÉTODOS

A identificação de um fator decisivo de desempenho em um WfMS, foi feita a partir da pesquisa desta tecnologia em livros, artigos e publicações obtidas na Internet, em páginas de empresas produtoras de WfMS, e através de contato direto com profissionais que pesquisam e desenvolvem trabalhos nesta área. Neste esforço, ficou claro que sistemas de *workflow* do tipo produção apresentam um cenário, onde diferenças de desempenho das atividades afetam processos, de maneira mais significativa que nos demais tipos. A partir deste ponto, ficaram definidas duas abordagens para a análise de desempenho. A primeira, levando em conta a influência do balanceamento de carga em sistemas cliente-servidor, correspondendo a aspectos puramente tecnológicos, e a segunda, considerando os fatores humanos que acarretam diferenças de rendimento em execução de processos.

No tocante aos aspectos tecnológicos do desempenho do sistema, o *hardware* e o *software* têm o mesmo efeito sobre qualquer executor humano de atividades. Como a atuação dos seres humanos apresenta diferenças individuais, a abordagem escolhida para o trabalho foi baseada na distribuição de atividades a executores humanos em sistemas de *workflow*. Quando uma atividade pode ser executada por mais de um participante, determinar o melhor candidato à mesma visa garantir ganho de desempenho.

A condução do trabalho ficou, desta forma, dependente da determinação dos fatores humanos que influem na produtividade, durante a execução de um processo em um WfMS. Buscou-se atender a duas aspirações, presentes em uma atribuição de responsabilidade, que são determinar a pessoa que produz o resultado com maior qualidade, em menor tempo, e garantir que se escolha alguém disponível para exercer tais habilidades. Foram então identificados dois fatores, a **aptidão pessoal** para a atividade, e a **disponibilidade** do ator para assumir uma responsabilidade específica. A identificação da **aptidão** do executor levou em consideração, para a atividade a ser distribuída, uma medida do grau de dificuldade da mesma, um grau de aceitação de execuções anteriores do ator para a atividade, e os dados de produtividade do ator, presentes no histórico de execução da atividade. Para a determinação da **disponibilidade** do ator, foi considerada uma combinação da prioridade da atividade no

processo, com o grau imediato de ocupação do ator. As variáveis que determinam os fatores citados, a serem utilizadas na inferência, com lógica nebulosa, para a avaliação do ator, são consideradas na seção 5.3 deste capítulo.

Para que a funcionalidade de atribuição de atividade ao ator mais qualificado pudesse ser comprovada, foram consideradas as hipóteses de se fazer apenas a modelagem de um WfMS e implementar o mecanismo de avaliação, e a de se desenvolver um sistema simplificado, simulando os padrões da WfMC. A decisão pela segunda hipótese, envolveu a especificação orientada a objetos, de todo o sistema, feita com a notação UML. Em uma especificação UML, os artefatos gerados são complementares, permitindo o entendimento dos aspectos estáticos e dinâmicos do sistema, dentro de um padrão aceito e consolidado.

O sistema WfFuzzy foi desenvolvido em linguagem Java, a partir da especificação UML e da conversão do código gerado em linguagem C, pela ferramenta de modelagem em lógica nebulosa. O ponto central de adequação da proposta a uma aplicação prática, está nas referências feitas aos padrões da WfMC.

5.1.2 TÉCNICAS

As técnicas para a implementação do WfFuzzy envolveram:

- O uso de um mecanismo de inferência baseado em lógica nebulosa;
- A adequação da solução proposta a um padrão mundialmente aceito para sistemas de *workflow*;
- A modelagem das variáveis utilizadas no mecanismo de inteligência artificial;
- A modelagem do sistema desenvolvido para a atribuição de atividades em um WfMS, envolvendo a especificação orientada a objetos e a definição da base de dados,
- A codificação do sistema em linguagem Java.

A comprovação dos resultados da inferência nebulosa foi feita através da submissão de 40 amostras de execuções de atividades ao sistema. As atividades fazem parte de 3 processos, compostos por 5 atividades cada um, em média. Foram considerados 13 atores, com direitos a executar as atividades, baseados em um dos 8 papéis funcionais registrados no sistema.

Foi utilizada uma ferramenta, para modelagem do problema em lógica nebulosa, chamada UnFuzzy, citada na seção a seguir. O uso de uma ferramenta assim se justifica, por facilitar a experimentação das soluções que se quer validar. A partir dela, foi feita a geração automática do código do mecanismo de inferência, em linguagem C. Este código foi convertido para a linguagem de programação Java. A geração das classes e dos métodos para o sistema WfFuzzy foi feita de forma manual, obedecendo a especificação de projeto, em UML, guiando a conversão de C para Java.

5.1.3 FERRAMENTAS

Ferramentas de modelagem e desenvolvimento, disponíveis comercialmente, foram utilizadas para criar os modelos de análise e projeto em UML, de lógica nebulosa e de dados, além da codificação em linguagem de programação. Estas são apresentadas a seguir.

Modelagem em UML

A ferramenta utilizada foi o *software* Rational Rose 2000, versão de demonstração, da empresa Rational Software Corporation.

Modelagem de Dados

O modelo de dados, apresentado no trabalho, foi desenvolvido com a ferramenta Erwin, versão ERX 3.5.2, da empresa Platinum Technology . A implementação do modelo foi feita através do *software* Microsoft Access 97, com o qual foram gerados os códigos das consultas na linguagem SQL, utilizados no algoritmo dos métodos das classes da aplicação.

Linguagem de Programação

O sistema WfFuzzy foi implementado na linguagem Java, utilizando o *software* JDK, versão 1.1.8 e o editor de programas TextPad, versão 4.4.2, da empresa Helios Software Solutions. O programa UnFuzzy gerou o código do mecanismo de inferência de cada variável de saída, em linguagem C.

Modelagem em lógica nebulosa

O desenvolvimento da modelagem, em lógica nebulosa, utilizou o programa UnFuzzy, versão 1.1, de distribuição gratuita, desenvolvido na Universidade Nacional da Colômbia e o sistema fuzzyTECH versão 5.31 Professional Demo, da empresa alemã INFORM GmbH. O primeiro, foi escolhido pela capacidade de geração de código em linguagem de programação e o segundo, devido aos recursos para modelagem visual interativa. As regras de inferência, resultantes da modelagem com o fuzzyTECH foram transcritas para o UnFuzzy. Ambos demonstraram o mesmo comportamento no momento da avaliação das variáveis, o que comprovou a veracidade dos resultados obtidos. A melhor análise gráfica, fornecida pelo fuzzyTECH, permitiu um ajuste mais refinado do comportamento das variáveis, e uma visão global da influência de cada variável no resultado final.

Desta forma, foi possível comprovar que a modelagem de uma solução, com lógica nebulosa, pode ser feita a partir da experimentação dos ajustes das entradas, baseada no conhecimento heurístico de quem modela.

O conhecimento sobre o comportamento de atores foi obtido a partir de entrevistas com profissionais que executam processos, em atividades comerciais, de gerência de ensino (UNIRONDON), e de fiscalização, em um conselho profissional (CREA/MT). A partir deste conhecimento, foi definido o comportamento esperado para cada saída, que determinou o ajuste das variáveis de entrada.

De modo a tornar a solução adaptável a qualquer contexto de execução de processos de negócio em um sistema de *workflow*, foi definido que os conjuntos nebulosos pudessem ser parametrizados a partir do WfFuzzy, de forma interativa.

No texto do trabalho e no sistema WfFuzzy, a **atividade** também foi referida como **tarefa**, que tem o mesmo significado.

5.2 MODELOS DO SISTEMA WfFuzzy

A simulação do WfMS compreende, de modo geral, a definição do processo, e a ação do motor do sistema sobre esta definição, executando o fluxo de trabalho e invocando a eleição dos atores, quando necessário. Nesta seção, são detalhados os modelos desenvolvidos para o sistema simulado de *workflow* e para o mecanismo de inteligência artificial.

5.2.1 MODELO DE DADOS

O modelo de dados para o protótipo de WfMS e para a aplicação com lógica nebulosa é exibido na Figura A.1, no Anexo A, na página 129. Para facilitar o entendimento, o modelo está dividido em áreas tracejadas, agrupando dados de definição do processo, dados organizacionais, dados de execução, e dados da aplicação em lógica nebulosa.

Os **dados organizacionais** determinam os empregados da empresa e os papéis existentes no organograma, que serão associados aos mesmos. Os **dados de definição do processo** estabelecem a identificação do processo, os papéis funcionais que ele abrange, os participantes responsáveis pelo desempenho destes papéis, as unidades de tempo utilizadas para redução a um fator comum de tempo de execução de atividades, os aplicativos invocados para apoiar a execução, as atividades que compõe o processo, os sub-processos e os dados da aplicação de lógica nebulosa. Os **dados da aplicação de lógica nebulosa** são estabelecidos e referenciados pela definição do processo, compostos pela variável nebulosa, os conjuntos nebulosos, e a associação destes com as atividades e as regras de inferência. O modelo de dados do WfFuzzy apresenta estruturas básicas, que dão apoio a uma execução simulada de processo. Não são detalhados os dados de transição entre atividades, nem os parâmetros para invocação de aplicativos, ou as pré e pós condições associadas às atividades e demais dados que fazem parte de um sistema adequado ao padrão da WfMC.

Esta definição de dados pode ser relacionada com o modelo abstrato da WfMC, como a seguir. Os **dados relevantes de *workflow*** no modelo abstrato estão relacionados aos **dados de execução do protótipo**, no tocante aos atores assinalados para as

atividades, às condições de transição, e aos dados de definição do processo, referentes às pré e pós condições das atividades. Os **dados da aplicação** dizem respeito aos **dados de lógica nebulosa**. Os **dados de controle** não são especificados, por serem apenas acessíveis ao WfMS específico e devido à simplificação atribuída ao WfFuzzy.

Cabe destacar, nos dados para inferência, que a **Variável**, representando a variável nebulosa, têm uma definição geral para o ambiente do sistema, podendo servir a qualquer **Processo**, sendo particularizada para cada **Tarefa**. Os parâmetros de variável, para todo o ambiente do *workflow*, são o nome e o número limite de conjuntos nebulosos. Para cada atividade, a Variável é configurada com os limites mínimo e máximo do seu universo de discurso, e com o número de intervalos de avaliação, para as regras de inferência. A associação de **Variável Tarefa** com **Conjunto Difuso**, que representa o conjunto nebuloso, resulta no **Conjunto**, que contém pontos, determinando os valores limite de cada conjunto envolvido nas regras de inferência. A **Regra**, relacionada ao conjunto, especifica o valor do modificador lingüístico e o tipo da variável envolvida, de entrada ou de saída. Contudo, **Regra** é uma estrutura destinada a considerações futuras, traçadas no capítulo 7, mas presente no modelo para fins de entendimento e complemento da proposta.

5.2.2 ESPECIFICAÇÃO ORIENTADA A OBJETO, EM UML

A especificação UML, utilizada para representar o WfMS que invoca a aplicação de classificação de atores, abrange aspectos estáticos, com **diagramas de caso de uso** e de **classes**, e aspectos dinâmicos, com **diagramas de seqüência**. No desenvolvimento do projeto foram simplificadas as funcionalidades limitadas à definição e execução de processos. O sistema deve definir dados básicos de processo e executar atividades de forma seqüencial, sem condições de junção, ou divisão de rotas, sem avaliar condições de transição, ou invocar aplicativos, que não o de avaliação de atores. A atenção está voltada para a distribuição de atividades com lógica nebulosa.

Para representar o comportamento do WfFuzzy, foram definidos os casos de uso relativos à definição e à execução do processo, constantes da Figura 5.1. A interação com o motor do *workflow* é uma abstração, para inserir o sistema WfFuzzy no contexto de um sistema de *workflow* real.

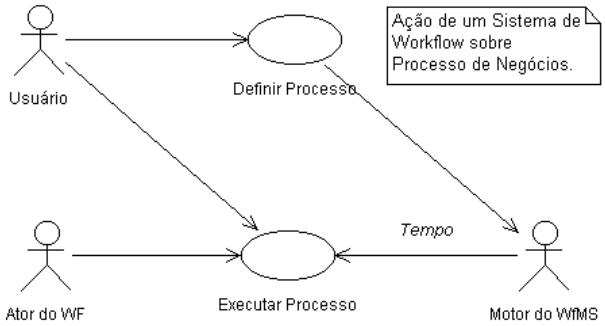

Figura 5.1 – Diagrama de Caso de Uso Principal do WfFuzzy

Na definição do processo, são especificados os dados de identificação do processo e de suas atividades. As variáveis nebulosas são parametrizadas neste caso de uso, e o único cálculo de inferência, representado nesta fase, é o do atributo **Grau de Dificuldade**, que faz parte da atividade, e é afetado quando se especifica uma nova quantidade de campos do formulário eletrônico que apoia a atividade, ou a quantidade de tempo para execução, externo ao uso do WfMS. O diagrama de caso de uso **Definir Processo** é mostrado na Figura B.1, no Anexo B, na página 130, e seu diagrama de seqüência correspondente, na Figura B.2, na mesma página.

O caso de uso **Executar Processo**, na Figura B.3, na página 131, envolve a instanciação do processo e de suas atividades, a invocação de atividades e a geração e execução dos itens de trabalho. A Figura B.4, na página 131, retrata o aspecto dinâmico da execução do processo. No caso de uso **Eleger Ator**, mostrado na Figura B.5, na página 132, são avaliadas as regras de inferência, para determinar a nota de cada ator candidato à atividade. O ator com maior nota é selecionado. Os casos de empate são possíveis, com maior possibilidade, no caso de não haver histórico de execução a ser avaliado e quando os atores apresentarem a mesma carga alocada de trabalho. Caso haja histórico de itens executados, um empate é uma situação resultante de inferência envolvendo todas as variáveis consideradas, e a escolha aleatória do ator, dentre os de mesma nota, é então justificada. A avaliação de cada ator candidato à atividade começa com a configuração das variáveis nebulosas da mesma, a partir da base de dados de conjuntos nebulosos. Os cálculos nebulosos ocorrem a partir da pesquisa das execuções do ator, para a atividade a ser distribuída, e são efetuados sobre as variáveis nebulosas, consideradas na seção 5.3. Na Figura B.6, na página 132, é representado o diagrama de

seqüência **Eleger Ator**, com a realização do caso de uso de mesmo nome. As interações que representam o caso **Calcular Nota**, um detalhamento do caso **Eleger Ator**, são mostradas na Figura B.7, na página 133. Após a eleição do ator mais qualificado, é feita a geração do item de trabalho para o mesmo. A Figura B.8, na página 133, mostra o diagrama de seqüência **Gerar Item de Trabalho**.

Na execução do processo é estabelecida a execução da tarefa, representada a partir da exibição dos itens de trabalho do ator. No diagrama da Figura B.9, na página 134, foi representada a invocação do aplicativo, devido à importância do formulário eletrônico envolvido no processo. Tanto iniciar, quanto completar a execução da atividade, podem ser feitos pelo ator ou pelo motor do *workflow*, dependendo da definição da mesma. No WfFuzzy, é considerada apenas a participação do ator humano. Neste caso, a execução da atividade se limita à anotação de **data e hora** de **início** e **fim** de execução. Isto se deve à facilidade de representar execuções distintas, criando histórico de execuções de forma produtiva, sem exigir alterações na data e hora do sistema operacional, durante a execução do sistema.

Diagrama De Classes do WfFuzzy

Uma vez que a WfMC não especifica uma estrutura para elementos de um WfMS, mas um meta modelo de definição e de interoperabilidade para *workflow*, o diagrama da Figura C.1, no Anexo C, na página 135, reflete as estruturas simplificadas, propostas neste trabalho. As classes relativas ao meta modelo de definição do processo e ao modelo de arquitetura abstrata da WfMC foram agrupadas em figuras com diferente cor de preenchimento, como forma de facilitar a visualização da adequação da solução proposta a um padrão existente. A exceção cabe à classe **Item de Trabalho**, pois sintetiza os itens a serem executados, a lista de trabalho e os dados de histórico do *workflow* em uma só estrutura.

As classes que compõem a solução proposta para distribuição de tarefas são representadas sem preenchimento, com fundo branco, no diagrama da Figura C.1. A classe **UnidadeTempo** representa o fator de tempo a ser aplicado para conversão a uma base única de tempos de execução de tarefas. Na implementação do WfFuzzy esta classe não foi considerada, a título de simplificação, mas foi mantida no modelo, para

facilitar o entendimento da flexibilidade a uma situação real. Todos os tempos para execução de tarefas são quantificados em horas.

A classe **Difusor** apresenta os parâmetros dos difusores, utilizados em cada variável de entrada, nas regras de inferência. A classe **Regra** agrega as variáveis de entrada e a variável de saída, que se combinam na regra de inferência. A classe **Conjunto** identifica os conjuntos nebulosos que agregam a super classe **Fuzzy**, da qual são herdadas as variáveis nebulosas para determinação da nota do ator. Estabelecendo a relação de herança entre Fuzzy e as classes que representam as variáveis nebulosas, foi possível definir, a uma só vez, os atributos que dimensionam os conjuntos nebulosos dos universos de discurso, e combinam as variáveis nas regras de inferência. As particularidades de cada variável resultante de inferência, são componentes das classes herdadas de Fuzzy.

5.2.3 MODELO DE INFERÊNCIA NEBULOSA

A determinação da nota atribuída ao ator do *workflow*, para a execução de uma atividade, é feita com base nas variáveis discutidas nesta seção, que podem ser configuradas dinamicamente, para acomodar diferentes atividades. Os conjuntos nebulosos são definidos na forma de trapezóide e triângulo, com dois ou três pontos significativos. Estes pontos podem ser entendidos como as cotas no eixo X, para as quais o valor de pertinência no eixo Y vale 0 ou 1. A Figura 5.2 mostra os tipos de função escolhidas neste trabalho, para todas as variáveis, onde são aplicadas estas considerações. O primeiro e o último conjuntos de cada variável, com forma **trapezoidal**, apresentam dois pontos significativos. Os intermediários, com formato **triangular**, têm três pontos, dois mínimos e um máximo, que representa o ponto de equilíbrio.

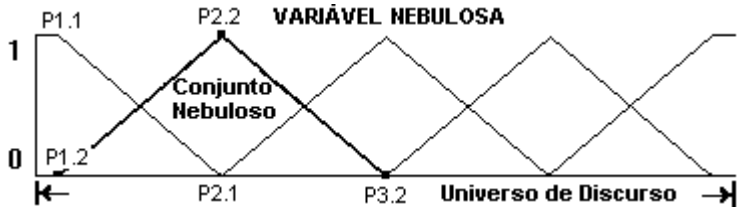

Figura 5.2 – Variável Nebulosa Padrão

Com a definição de uma base de dados, os conjuntos nebulosos podem ser alterados para adaptação a cada variável, permitindo que o comportamento inteligente do sistema possa ser configurado. As alterações permitem que se defina o número de conjuntos e os seus limites, dentro do universo de discurso. Para reduzir o número de conjuntos de uma variável, deve-se atribuir valor nulo aos pontos do conjunto eliminado e os conjuntos restantes deverão ocupar seu espaço no universo de discurso, a partir de uma redefinição de seus pontos na base de dados. As dimensões dos conjuntos podem ser alteradas e, se fossem definidos quatro pontos, seria possível ocorrer variações entre triângulo e trapezóide para um mesmo conjunto. Neste trabalho os conjuntos foram definidos com três pontos, pelas razões explicadas na seção 4.3 do capítulo 4.

5.3 VARIÁVEIS NEBULOSAS

As variáveis que definem a nota do ator para a execução da atividade são mostradas na Figura 5.3. O símbolo na figura, ao lado do nome de cada variável, foi estabelecido para a classificação em variáveis de entrada, com símbolo à esquerda, de saída, com símbolo à direita, e intermediárias, com símbolos de ambos os lados. As variáveis intermediárias são variáveis de saída de um cálculo nebuloso, que servem de entrada para o cálculo de outra variável. Nas figuras da seção 5.3.1, que detalham cada variável calculada em lógica nebulosa, a escala na parte inferior classifica os valores da variável e permite melhor entendimento do gráfico resultante.

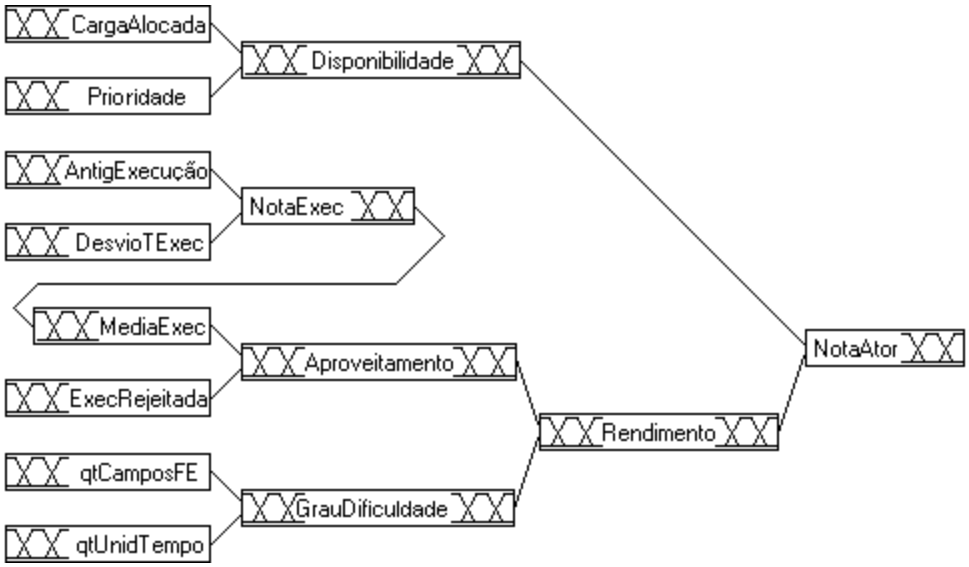

Figura 5.3 – Modelo de Inferência Nebulosa

Cada variável pode ser configurada pelo WfFuzzy, através da definição dos pontos dos seus conjuntos nebulosos, podendo ser adaptada para as diferentes situações que uma definição de processo exigir. O comportamento das variáveis comentadas a seguir pode, desta forma, variar de acordo com a definição dos conjuntos nebulosos e das regras de inferência das quais elas participam. Alterações no valor do universo de discurso de uma variável não mudam as regras de inferência, uma vez que os conjuntos são alterados na mesma proporção. A seguir, são detalhadas as variáveis utilizadas para a eleição do ator. Os detalhes da distribuição dos pontos dos conjuntos, nos universos de discurso de cada variável, podem ser vistos na Tabela 6.1, na página 107, mas na definição de cada variável é mostrada uma tabela com os limites mínimo e máximo de cada conjunto. As variáveis de entrada e de saída tem escala linear. As figuras que representam o comportamento de cada variável de saída, em função das duas variáveis de entrada, apresentam uma escala vertical para a saída e duas escalas horizontais, em perspectiva, para as entradas.

5.3.1 GRAU DE DIFICULDADE DA TAREFA

O grau de dificuldade da atividade, ou tarefa, exerce influência no desempenho do ator, dependendo de sua aptidão para executá-la. Portanto, os fatores considerados

para a determinação deste parâmetro dizem respeito à manipulação da tarefa pelo ator. As variáveis de entrada para o cálculo do **Grau de Dificuldade da Tarefa**, definidas neste trabalho, são a **quantidade de campos** a serem preenchidos no formulário eletrônico que apoia a tarefa, e a **quantidade prevista de tempo demandado para execução**, externo ao gasto com recursos do WfMS. A quantidade de campos do formulário foi estabelecida a partir de troca de idéias com o prof. Tadeu Cruz, profissional da área de automação de processos. A Figura 5.4, exibe o comportamento do grau de dificuldade, em função da variação das variáveis de entrada. Sua variação cobre ao valores de zero (0) a dez (10), detalhada na Tabela 5.1. O grau de dificuldade é armazenado na fase de definição do processo e é utilizado na determinação do rendimento do ator.

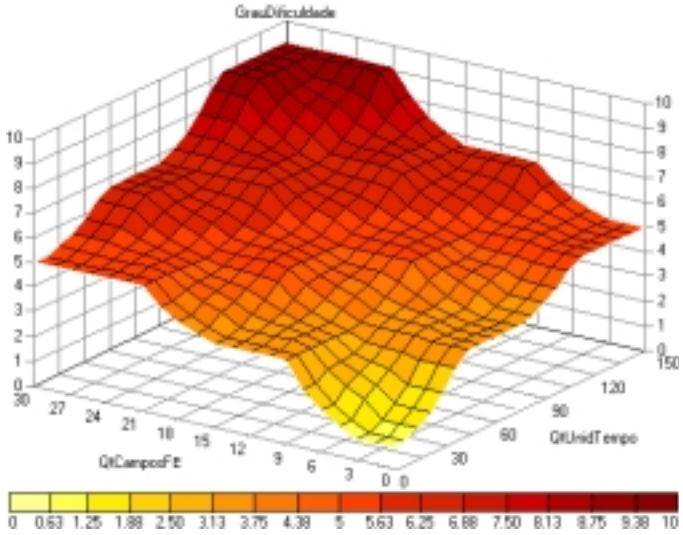

Figura 5.4 – Grau de Dificuldade da Atividade

Tabela 5.1 – Conjuntos do Grau de Dificuldade da Atividade

GrauDificuldade	Variável Lingüística
0 A 3	Mínimo
1 a 5	Baixo
3 a 7	Médio
5 a 9	Alto
7 a 10	Máximo

Quantidade de Campos do formulário Eletrônico – QtCamposFE

Processos em um WfMS envolvem documentos eletrônicos na execução de atividades, do mesmo modo que os documentos em papel estão presentes em processos manuais. Quando o documento é um formulário, os campos a serem preenchidos no mesmo demandam esforço do ator. A métrica para a determinação deste esforço deve levar em conta uma unidade genérica, que permita sua aplicação aos diferentes tipos de formulário, em um ambiente de trabalho. A quantidade de campos é estabelecida no momento da definição do processo. O valor zero indica que não há campos de formulário eletrônico envolvido na tarefa, ou que não há sequer um formulário apoiando a tarefa. A medida da quantidade de campos pode ser determinada com base na técnica de análise de pontos de função (FPA) de um sistema de informação. A análise por ponto de função – FPA provê medidas, independentemente de tecnologia de implementação, para apoiar a análise de produtividade e qualidade e para estimar o tamanho do *software*, fornecendo um fator de normalização para comparação de *software*. A aplicação de FPA permite medir o tamanho de uma aplicação ou de um projeto e, quando utilizada em conjunto com outra medida, pode determinar o nível de produtividade da equipe, o esforço de desenvolvimento de *software*, entre outros. Seu uso pode ser estendido para medir o tamanho de um documento. Os campos, neste caso, determinam o tamanho do documento. Esta definição é estabelecida por um profissional experiente na análise de processos de negócio.

Neste trabalho, a quantidade de campos foi determinada a partir das amostras de documentos, obtidos nas organizações citadas na seção 5.1.3. A partir destas amostras, foi definido o limite do universo de discurso, de zero (0) a trinta (30) campos, e de cada um dos conjuntos, mostrado na Tabela 5.2.

Tabela 5.2 – Quantidade de Campos do Formulário Eletrônico

QtCamposFE	Variável Lingüística
0 A 8	Mínima
1 a 15	Baixa
8 a 22	Média
15 a 29	Alta
22 a 30	Máxima

Quantidade de Tempo Externo ao WfMS para a Tarefa – QtUnidTempo

Além do tempo gasto com o uso de recursos do sistema de *workflow*, tarefas demandam tempo para planejamento, tomada de decisão, reflexão, autorização, análise de resultados, e subsídios para a preparação das informações necessárias à sua execução. O tempo despendido nestas atividades é adicional ao investido em uso dos recursos do sistema e portanto, definido como externo ao WfMS. O tempo externo é um fator mais significativo para avaliar a dificuldade de desempenho da tarefa, pois é mais passível de grandes variações. O tempo deve ser convertido, baseado em um fator para unificação da unidades, de forma a permitir que seja estabelecida a parametrização desta variável na definição do processo. A especificação do universo de discurso e dos conjuntos nebulosos desta variável, deve levar em conta valores de tempo de execução, advindos de amostras do histórico de execução de cada atividade, e dos patamares esperados como tempo padrão de produtividade, na organização onde o sistema opera. Este tempo é definido por especialistas, na análise da execução do negócio. A partir disto, uma tarefa pode ser classificada, em relação a esta variável. Os conjuntos para o universo de discurso desta variável, de zero (0) a cento e cinqüenta (150) unidades de tempo, são mostrados na Tabela 5.3.

Tabela 5.3 – Quantidade de Unidades de Tempo Externo

QtUnidTempo	Variável Lingüística
0 A 40	Mínima
5 a 75	Baixa
40 a 110	Média
75 a 145	Alta
110 a 150	Máxima

5.3.2 DISPONIBILIDADE DO ATOR

A **disponibilidade do ator**, em assumir a execução de uma atividade, foi definida como a relação entre a **carga de trabalho alocada** ao ator e a **prioridade da tarefa** a ser distribuída. A disponibilidade é atribuída ao ator como um índice que define o ator como candidato à execução da atividade. Uma baixa disponibilidade significa que o ator está sobrecarregado, para a prioridade requerida. O comportamento da **Disponibilidade**, em função das variáveis de entrada, pode ser observado na Figura 5.5.

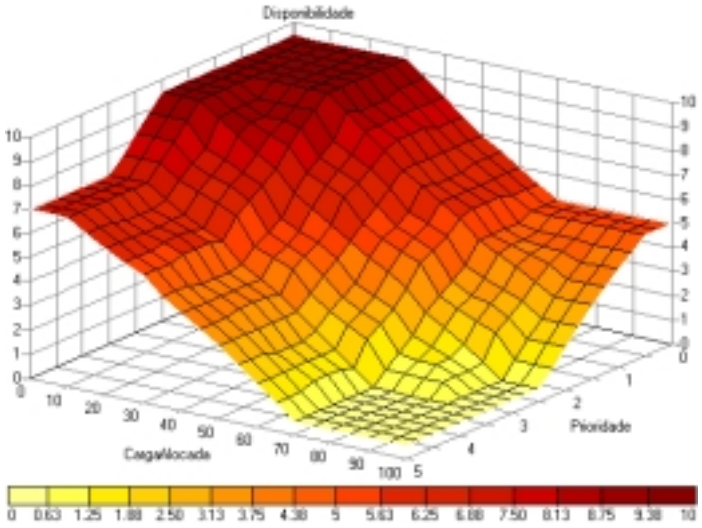

Figura 5.5 – Disponibilidade do Ator

Na Tabela 5.4, são mostrados os conjuntos nebulosos para a variável Disponibilidade do Ator.

Tabela 5.4 – Conjuntos da Disponibilidade do Ator

Disponibilidade	Variável Lingüística
0 A 3	Mínima
1 a 5	Baixa
3 a 7	Média
5 a 9	Alta
7 a 10	Máxima

Carga de Prioridade Alocada – CargaAlocada

Para determinar uma medida de carga de trabalho, foi definida uma variável que pudesse expressar, de forma quantitativa, o volume das atividades que devem ser executadas. Estas atividades estão entre os itens de trabalho ainda não executados, ou em execução. A forma de medir este volume de trabalho foi relacionada à urgência requerida por cada atividade não executada, determinada pela prioridade da mesma. A cada prioridade é atribuída uma nota, que pontua o grau de urgência para a execução da atividade. A atribuição da nota deve ser feita na ordem inversa do valor de prioridade que ela deve pontuar. Assim, valores numéricos baixos indicam prioridades elevadas, como pode ser observado na Tabela 5.5. Desta forma, foi criada a variável de entrada **CargaAlocada**, que é a medida da ocupação do ator, feita a partir das atividades pendentes, constantes da sua lista de trabalho. Ela é determinada pelo somatório da nota correspondente à prioridade de cada atividade.

Tabela 5.5 – Nota da Carga em Função da Prioridade

Prioridade	Variável Lingüística	Nota de Carga Pendente
0	Máxima	5
1	Alta	4
2	Média	3
3	Baixa	2
4	Mínima	1

O comportamento da **disponibilidade do ator**, exibido na Figura 5.5, evidencia que atividades só passam a ser atribuídas a atores com carga alocada elevada, a partir de prioridades elevadas, enquanto que para atores com carga leve, a disponibilidade elevada ocorre a partir de prioridades reduzidas. Isto faz com que o sistema não sobrecarregue inutilmente os atores já ocupados, mas lhes atribua pontuação que os torna candidatos, apenas quando o nível de urgência for alto.

Prioridade da Tarefa

A tarefa possui uma prioridade, informada na definição do processo, que determina a importância da sua execução no processo. Os valores lingüísticos desta variável são mostrados na Tabela 5.5. A prioridade de tarefa pode não refletir a real necessidade da distribuição. Se a tarefa tiver baixa prioridade e o processo estiver atrasado, o WfMS deve poder controlar sua alocação, independentemente do mecanismo de eleição do ator. Um WfMS real, deve também poder alterar a prioridade de uma tarefa por um fator, definido uma única vez para todo o processo, utilizado quando se deseja acelerar ou reduzir a execução, em função de atrasos ou de liberação de recursos para outras instâncias de processos.

Um ator com carga de trabalho muito elevada passa a estar disponível apenas quando a prioridade da tarefa for elevada, pois o fato de contar com muitos itens em sua lista de trabalho pode ser um indicativo de sua capacidade de execução, que justifica poupá-lo de sobrecarga com tarefas de menor importância. De outra forma, a quantidade de itens pendentes em sua lista, pode determinar sua baixa capacidade de resolução. A questão de penalizar um bom executor com carga extra de trabalho é um aspecto que deve ressaltar o desequilíbrio de produtividade na organização e ser objeto de reestruturação e qualificação de pessoal, a ser administrado pela gerência do processo.

5.3.3 NOTA E MÉDIA DE EXECUÇÃO

Outro fator que determina a aptidão do ator é a avaliação do seu desempenho na execução de instâncias da atividade a ser distribuída, a partir do seu histórico de execuções. Avaliar este desempenho consiste em determinar uma **nota de execução**, para cada item executado pelo ator e, ao final das ocorrências no histórico, determinar a

nota resultante da **média** aritmética das amostras. Para cada execução, a nota que a qualifica é quantificada a partir da variação entre o tempo realmente gasto para executar, em relação ao tempo padrão para tal, estabelecido para a atividade no momento da definição do processo. Esta variação foi denominada de **Desvio no Tempo de Execução** da atividade. De forma a evitar que um desvio de tempo de execução, muito baixo ou muito alto, perpetue um bom ou mau resultado esporádico, foi considerada a **antigüidade da execução**, como fator atenuante do desempenho. Quando o ator não tiver histórico de execuções para a atividade considerada, é atribuída a ele, pelo sistema, a nota de execução média (5). A Figura 5.6 ilustra o comportamento da variável **nota de execução** do ator para a atividade e, como a **média de execução** é a média destas notas, seu comportamento é o mesmo da nota individual.

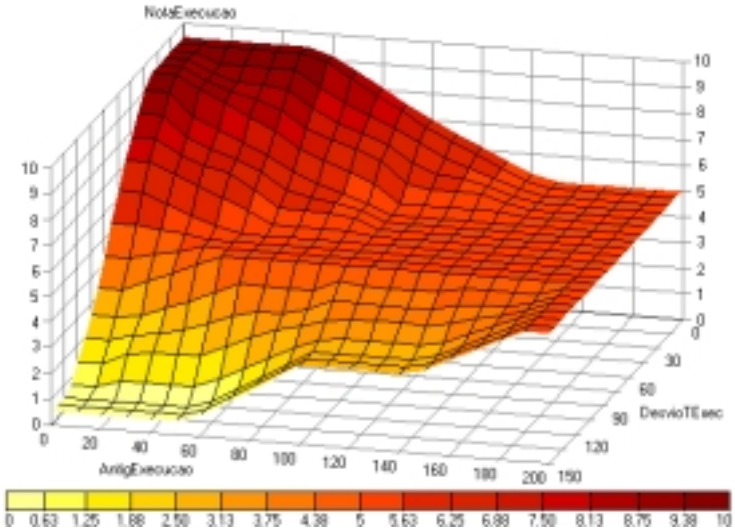

Figura 5.6 – Nota e Média de Execução do Ator

A Tabela 5.6 mostra os valores dos conjuntos nebulosos de Nota e Média do ator.

Tabela 5.6 – Conjuntos da Nota e Média de Execução do Ator

NotaExecução e MediaExecução	Variável Lingüística
0 A 3	Mínima
1 a 5	Baixa
3 a 7	Média
5 a 9	Alta
7 a 10	Máxima

Desvio do Tempo de Execução

Para cada atividade que compõe um processo, é definido um tempo padrão para sua execução, o que é responsabilidade de um especialista em análise de processos de negócio. O tempo real da execução de cada instância da tarefa pode variar, para mais ou menos, em relação a este tempo padrão. A variável DesvioTempoExec representa, em termos percentuais, a amplitude da variação, para a execução da tarefa. Um tempo menor que o padrão é um indicador de alta produtividade e, por conseguinte, a baixa produtividade é apontada por uma execução além do tempo estabelecido. Diversos fatores influem para que ocorra este desvio de tempo, como a falta de uma informação específica, a indisponibilidade do sistema durante um período, dentre outras, alheias à vontade do executor. Os WfMS estão inseridos em ambientes, onde um dos requisitos é a alta disponibilidade do sistema como um todo. Desta forma, desvios de execução, advindos de falhas no sistema, são minimizados e podem ser tratados como exceções. O que de fato interessa nesta variável é o desvio advindo de fatores humanos. Se estes forem devidos a outros fatores, que não os relativos ao executor da tarefa, ainda servem como indicadores da capacidade do ator na resolução de problemas. Importa que, na definição do processo, seja estabelecido um tempo padrão, condizente com a realidade do ambiente de negócio. A determinação do desvio deve, assim como para as demais variáveis, levar em consideração a realidade e os objetivos do ambiente onde o processo está inserido.

Para uma tarefa a ser distribuída, esta variável é obtida a partir das execuções de instâncias da tarefa, recuperadas a partir dos itens de trabalho executados.

O tempo padrão é estabelecido na fase de definição do processo. O desvio considerado nas execuções, para o trabalho, ficou estabelecido entre os limites de –50%, correspondente à metade do tempo padrão, e 100%, o dobro do tempo padrão. Neste caso, o tempo padrão de execução situa-se a um terço da escala, no ponto X=50%. Uma execução neste tempo pode ser considerada como muito boa, ou desejável. Para evitar trabalhar com valores negativos no universo de discurso, o desvio padrão é fixado como referencial para desvio em menor tempo, o que define o universo de discurso das amostras citadas entre 0%, para metade do tempo e 150%, para o dobro do tempo. A Tabela 5.7 ilustra os conjuntos para esta variável.

Tabela 5.7 – Conjuntos do Desvio do Tempo de Execução

DesvioTExec	Variável Lingüística
0 A 40	Mínimo
10 a 70	Baixo
40 a 100	Médio
70 a 130	Alto
100 a 150	Máximo

Para se calcular o desvio de tempo de execução, deve-se recuperar a unidade de medida de tempo da atividade. Com a unidade, faz-se a leitura do intervalo entre o início e o fim da execução, calcula-se quantas unidades de tempo foram gastas, e pode-se então determinar o desvio percentual do tempo padrão.

Antigüidade da Execução

A variável **AntigExecucao** considera o tempo em dias, decorrido da execução de uma atividade, até o momento da distribuição de outra instância da mesma atividade. Ela se justifica, por representar o tempo como atenuante dos resultados de um evento. Seja o efeito do desvio positivo, ou negativo, com o decorrer do tempo ele é amenizado,

para a determinação de um coeficiente de produtividade do ator que o gerou, como forma de evitar a perpetuação de um resultado. Os limites dos conjuntos de **AntigExec** são mostrados na Tabela 5.8.

Tabela 5.8 – Conjuntos do Antigüidade de Execução

AntigExec	Variável Lingüística
0 A 16	Mínima
3 a 30	Baixa
16 a 44	Média
30 a 57	Alta
44 a 60	Máxima

5.3.4 APROVEITAMENTO DO ATOR

O **aproveitamento** de um ator, no desempenho de instâncias da atividade a ser distribuída, tem importância como um fator complementar para a medida da qualidade de seu trabalho. Uma vez que a nota atribuída às suas execuções é baseada no tempo gasto para completar as atividades, mostrou-se importante avaliar o grau de satisfação com as execuções da atividade. Para tanto, foi definida uma variável que registra a **quantidade de execuções rejeitadas**, sob responsabilidade de uma pessoa que controle a qualidade do processo da execução. Esta variável serve ainda para equilibrar o efeito do **desvio do tempo de execução**, em relação ao tempo padrão definido para a atividade. Um desvio que indique aumento de produtividade, pode esconder o fato da atividade ter sido mal executada. Levando-se em consideração o número de execuções mal sucedidas do ator, pode-se equilibrar a rapidez com a qualidade final da execução. O comportamento do **Aproveitamento,** em função das variáveis **Média de Execução** e **Execuções Rejeitadas**, pode ser observado na Figura 5.7.

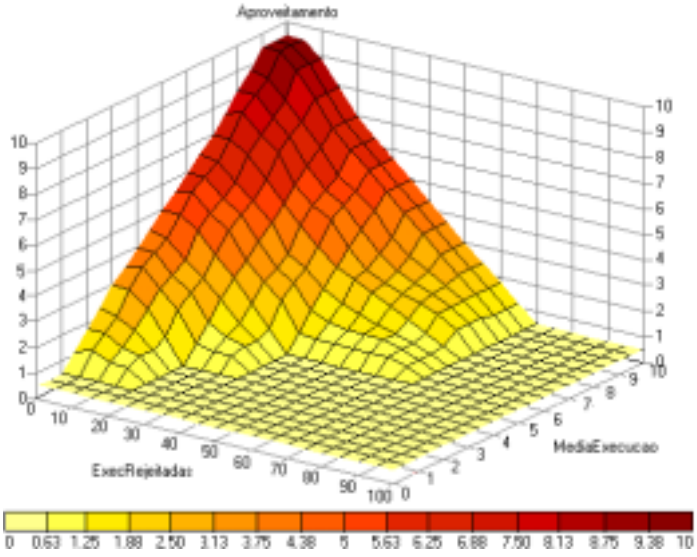

Figura 5.7 – Aproveitamento do Ator

Os conjuntos que definem o aproveitamento do ator são definidos pelos parâmetros mostrados na Tabela 5.9.

Tabela 5.9 – Conjuntos do Aproveitamento do Ator

Aproveitamento	Variável Lingüística
0 A 3	Mínimo
1 a 5	Baixo
3 a 7	Médio
5 a 9	Alto
7 a 10	Máximo

Quantidade de Execuções Rejeitadas – ExecRejeitadas

Como forma de medir a qualidade do trabalho, foi apenas considerado o fato de cada execução da tarefa ter ou não sido rejeitada, por um responsável pela avaliação da qualidade do trabalho. Outra forma de medir a qualidade seria estabelecer um grau de aceitação para cada execução e relacioná-lo à nota de execução, comentada na seção

5.3.3. Desta forma, ter-se-ia um controle mais apurado da qualidade do trabalho do ator. Contudo, a necessidade de pontuar a execução poderia causar impacto no desempenho do processo. Assim sendo, uma solução seria a criação de um mecanismo automático de avaliação da execução, o que levaria a um estudo fora do escopo deste trabalho, uma vez que o que se propõe é comprovar uma proposta que, a partir das variáveis de entrada, possa realizar uma tomada inteligente de decisão. O resultado da avaliação realizada pelo mecanismo ora citado, constitui-se em uma variável de entrada que, para simplificação, foi substituída por um grau de rejeição das execuções, determinado pela quantidade de execuções rejeitadas.

O identificador de execução rejeitada para o ator, é um atributo recuperado a partir dos itens de trabalho executados. A quantidade de execuções rejeitadas registra o número de tarefas encerradas, que não foram completadas, ou que foram recusadas, por responsabilidade única do ator, sendo definida pelo somatório de todas as execuções neste estado, feitas pelo ator sendo avaliado.

Esta variável têm um peso elevado, atuando de forma punitiva no resultado do aproveitamento, quando seu valor é alto.

Os conjuntos que definem a variável Media de Execuções são mostrados na Tabela 5.6 e os que definem a Quantidade de Execuções Rejeitadas, na Tabela 5.10.

Tabela 5.10 – Conjuntos da Quantidade de Execuções Rejeitadas

ExecRejeitadas	Variável Lingüística
0 A 1	Mínima
0 a 2	Baixa
1 a 4	Média
2 a 7	Alta
4 a 10	Máxima

5.3.5 RENDIMENTO DO ATOR

Uma vez que o Aproveitamento do Ator leva em consideração o histórico de execuções das atividades, pontuando com base na rapidez com que o ator executou atividades e na qualidade do seu trabalho, considerou-se necessário relacionar o resultado obtido à dificuldade estabelecida para a atividade, de forma a determinara aptidão do ator em executá-la. A variável **Rendimento do Ator** estabelece a relação entre as variáveis **Aproveitamento** e **Grau de Dificuldade da Tarefa**.

Nesta relação, os conjuntos nebulosos foram parametrizados, de forma a retratar o fato de que atividades com dificuldade mínima, podem conferir aproveitamento considerável a atores com aproveitamento baixo e não elevam muito o rendimento dos que tem excelente aproveitamento. No outro extremo, para atividades consideradas de máxima dificuldade, atores com baixo aproveitamento são muito penalizados em seu rendimento, enquanto que há grande valorização dos que tem alto aproveitamento, como pode ser observado na Figura 5.8. Desta forma, busca-se equilibrar a alocação de talentos às dificuldades apresentadas.

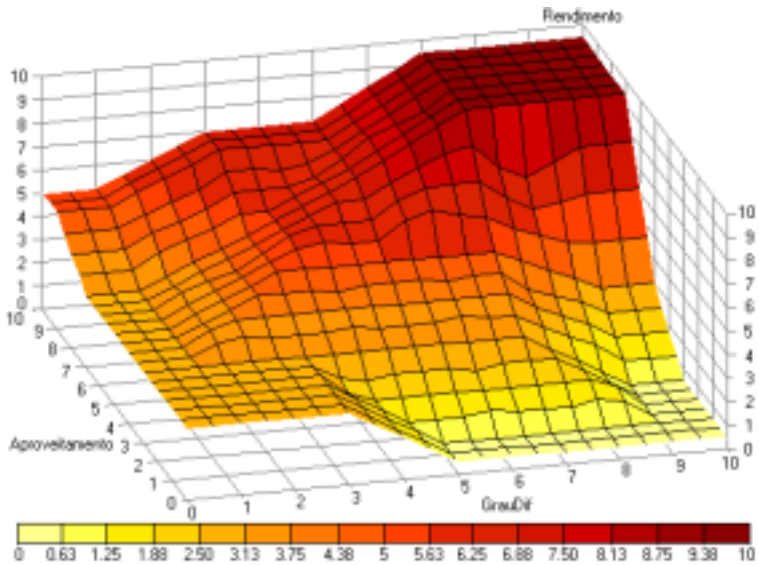

Figura 5.8 – Rendimento do Ator

A Tabela 5.11 mostra a definição dos conjuntos para o **Rendimento do Ator**.

Tabela 5.11 – Conjuntos do Rendimento do Ator

Rendimento	Variável Lingüística
0 A 3	Mínimo
1 a 5	Baixo
3 a 7	Médio
5 a 9	Alto
7 a 10	Máximo

5.3.6 NOTA DO ATOR

Toda atividade deveria ser destinada ao ator com maior rendimento, mas este pode estar sobrecarregado em relação a outros atores. Por outro lado, um ator mais livre pode apresentar baixo rendimento em seu trabalho. A variável **Nota do Ator** combina seu **Rendimento** com a sua **Disponibilidade**, de forma a equilibrar a alocação de uma atividade a um ator que possa executá-la com qualidade e a tempo, conforme pode ser observado na Figura 5.9.

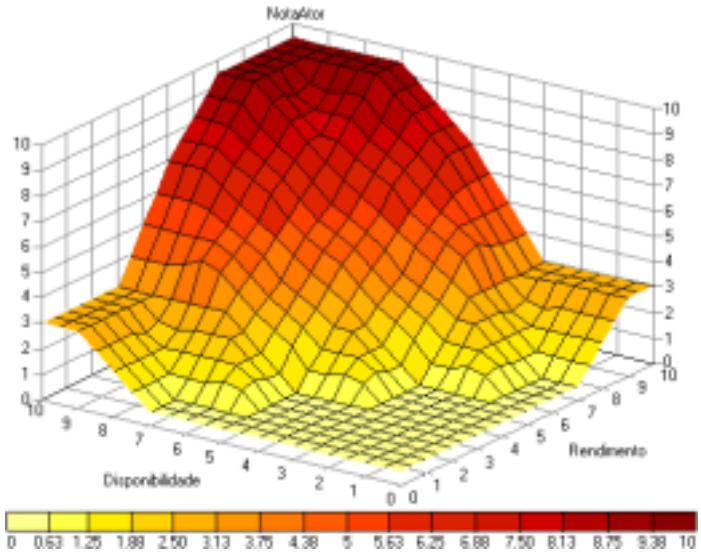

Figura 5.9 – Nota do Ator

A tabela 5.12 mostra os conjuntos definidos para a Nota do Ator.

Tabela 5.12 – Conjuntos da Nota do Ator

Nota	Variável Lingüística
0 A 3	Mínima
1 a 5	Baixa
3 a 7	Média
5 a 9	Alta
7 a 10	Máxima

5.4 CONSIDERAÇÕES

- Quando um número de execuções rejeitadas reduz em muito a nota do ator, ele fica praticamente indisponível, a não ser para atividades extremamente simples, com baixo grau de prioridade. Desta forma, a alteração dinâmica do estado de disponibilidade é possível, caso o ator realize a contento as atividades de pequena importância. Além disto, é possível alterar a situação do ator, suspendendo-o para qualquer execução.

- As amostras feitas para este trabalho, levaram em consideração uma determinada configuração das variáveis. Outras definições de pontos nos conjuntos irão gerar resultados diferentes, com comportamento diverso do comentado na seção 5.3.

- Um WfMS pode permitir que o administrador do fluxo atribua uma atividade a um ator específico, mesmo que ele não desempenhe o papel funcional necessário no processo. A segurança, neste caso, fica a cargo das regras para este tipo de exceção e do acesso à base de dados do sistema.

- Um parâmetro que identifica se a atividade é passível de execução por mais de um ator poderia ser especificado, ampliando a flexibilidade do WfFuzzy.

- Neste trabalho, a relevância é para atores humanos e individuais, se bem que a política se aplica, sem modificações de abordagem para cálculo de nota, igualmente para equipes. A diferença está na montagem das listas de trabalho. Para equipes, o trabalho alocado não aparece na lista dos seus componentes, mas na lista de um super ator, representante da equipe. Ele pode atribuir partes da tarefa a todos os atores da equipe. O tratamento de componentes de equipes, nestes casos, pode ser feito como uma extensão do tratamento dado a indivíduos.

- A lista de trabalho, que contém as atividades destinadas ao ator, foi desenvolvida de forma a ressaltar os aspectos importantes abordados na distribuição de atividades. A estrutura de uma lista de trabalho, destinada a atender grande número de sistemas de *workflow*, deve conter o maior número viável de campos, de forma a se adequar às necessidades de cada modelo.

- Para este trabalho, as atividades são distribuídas entre as listas dos atores, e lá permanecem, mesmo depois de executadas ou canceladas. O que diferencia o estado das atividades é um identificador de situação, que pode assumir os estados **A** (Ativa), **C** (Completada) e **E** (Executando).

- A interoperabilidade entre WfMS distribuídos, utilizando o mecanismo de distribuição de atividades proposto neste trabalho, irá requerer das organizações participantes, entre outras, a mesma política de definição das variáveis citadas.

- As informações para determinação da distribuição de atividades entre atores, constam das **informações relevantes de *workflow***, do modelo abstrato da arquitetura da WfMC.

- A ordem de execução das atividades é feita a partir do seu identificador numérico, que permite a inserção de valores entre dois inteiros. Desta forma, se uma nova atividade tiver que ser inserida no processo, ela pode ocupar sua ordem de execução, sem que seja necessário alterar outras atividades.

5.5 IMPLEMENTAÇÃO DO WFFUZZY

O sistema WfFuzzy apresenta as funcionalidades de definição do processo, geração e execução dos itens de trabalho. Na Figura 5.10 é exibida a janela de abertura do WfFuzzy, com as funcionalidades citadas.

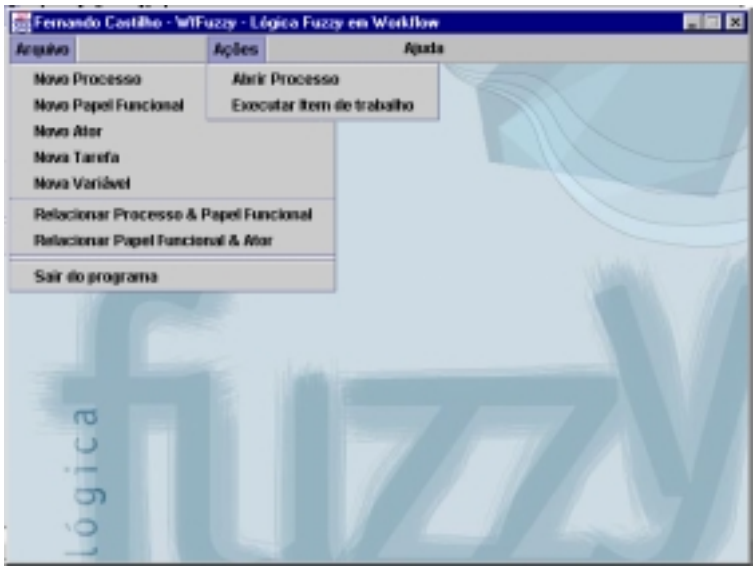

Figura 5.10 – Janela de Abertura do WfFuzzy

Os cadastros do sistema compreendem os elementos da definição do processo, mostrados na Figura 5.11. A definição do processo é feita a partir do registro dos dados de **Processo**, **Papel Funcional**, **Ator**, **Papéis do Processo**, **Papéis do Ator** e **Tarefa**. Papéis do Processo relaciona os papéis funcionais que podem ser executados no processo e Papéis do Ator relaciona Papéis do Processo com Ator, definindo **quem** pode fazer **o que** no processo.

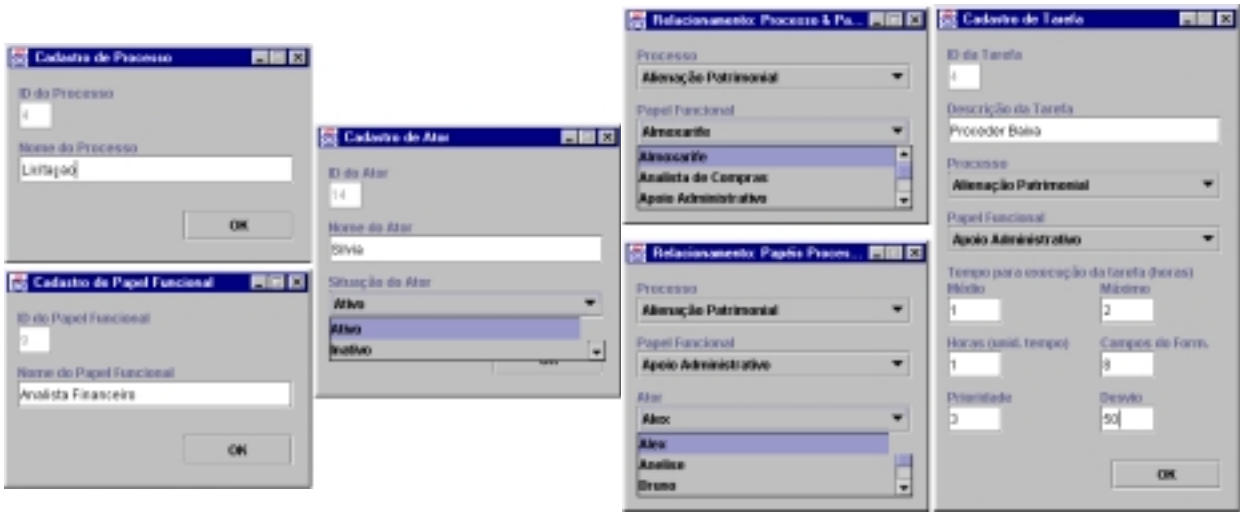

Figura 5.11 – Elementos da Definição do Processo

Na definição da Tarefa, é assinalado o seu **tempo padrão** para execução, que serve de base para medida da variável de entrada DesvioTExec, para o cálculo da nota de execução. O campo **Desvio**, na Figura 5.11, serve para referenciar valores negativos de desvio, como comentado na seção 5.4.3. Os campos **Horas** e **Campos do Formulário** registram as variáveis de entrada para o cálculo do Grau de Dificuldade da Tarefa, citado na seção 5.4.1 e o campo **Prioridade** registra a prioridade padrão da Tarefa, usada como referência de prioridade para a geração do item de trabalho. A prioridade do item de trabalho é utilizada no cálculo da Disponibilidade do Ator, citada na seção 5.4.2.

As variáveis nebulosas são cadastradas na definição do processo, conforme a janela da Figura 5.12. No registro da variável, é definido o número limite de conjuntos nebulosos que esta variável pode assumir. Este é um parâmetro do comportamento das variáveis no ambiente do *workflow*. Uma questão a ser considerada é a definição de um limite de conjuntos nebulosos para cada tarefa, como forma de atribuir maior especialização da definição do processo.

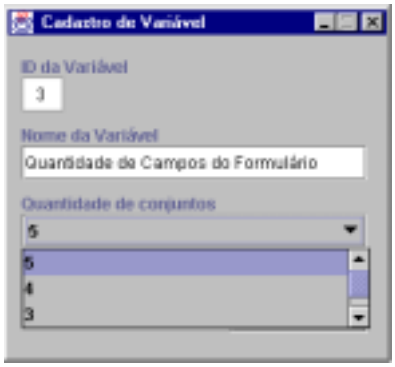

Figura 5.12 – Dados Básicos da Variável Nebulosa

A definição do processo é concluída através do registro das variáveis de cada atividade, invocada ao final do cadastramento dos dados básicos da Tarefa. O registro da Variável da Tarefa, na janela mostrada na Figura 5.13, é o momento da configuração dos conjuntos nebulosos para cada atividade no processo. Isto é feito registrando-se os pontos das funções de pertinência de cada conjunto nebuloso.

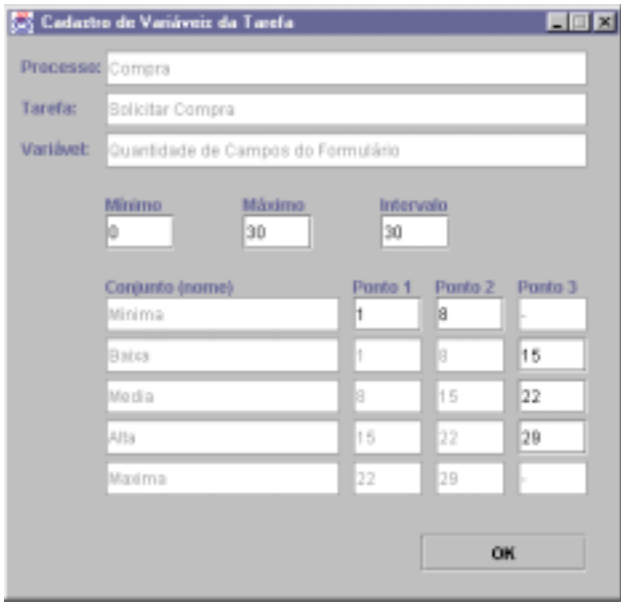

Figura 5.13 – Janela da Variável da Tarefa

Os pontos representam os **valores significativos**, quando ocorrem os valores **máximo** e **mínimo** da função de pertinência. Os conjuntos inicial (Mínima) e final (Máxima), na Figura 5.13, têm apenas dois pontos, pois são **trapezoidais**. Os demais têm função de pertinência **triangular**, com três pontos. No WfFuzzy, apenas são

registrados os pontos destacados em negrito na Figura 5.13. Os demais são completados automaticamente, em virtude do tipo de variável nebulosa utilizada, comentado na seção 5.3., o que facilita sua operação.

A título de exemplo, a Figura 5.14 exibe a variável nebulosa Quantidade de Campos do Formulário Eletrônico (qtCamposFE), com as funções de pertinência definidas a partir dos pontos informados na Figura 5.13.

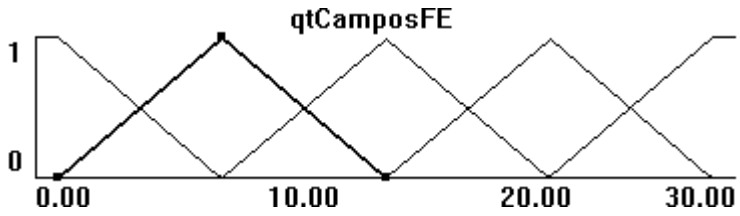

Figura 5.14 – Variável Quantidade de Campos do Formulário Eletrônico

Ao final da definição das variáveis para a tarefa, o sistema WfFuzzy gera o resultado do Grau de Dificuldade da Tarefa, para então terminar a definição da mesma no processo. Quando for registrada a última tarefa, estará terminada a definição do processo no sistema. A Figura 5.15 mostra o resultado do cálculo do Grau de Dificuldade para uma tarefa, utilizada na análise de resultados, comentada no capítulo 6.

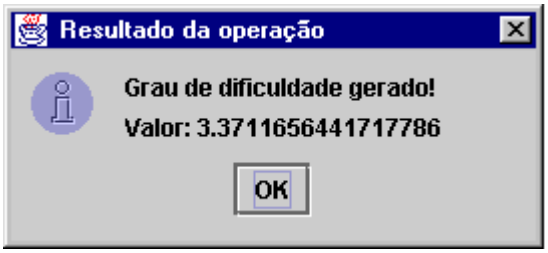

Figura 5.15 – Resultado do Cálculo do Grau de Dificuldade da Tarefa

Executar um processo no WfFuzzy consiste em uma atividade manual de seleção e abertura do mesmo. O comportamento padrão de um WfMS é representado, neste sistema, a partir da recuperação da tarefa ativa, ou seja, a próxima a ser executada no roteamento do processo, seguida da invocação da eleição do ator e da geração do item

de trabalho para o ator selecionado. A Figura 5.16 mostra a opção do WfFuzzy para abrir o processo.

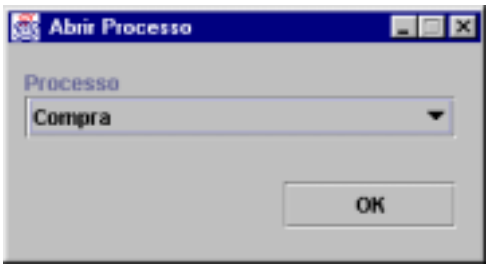

Figura 5.16 – Janela para Executar o Processo

Na geração dos itens, quando há mais de um ator qualificado para a tarefa, o sistema executa a eleição do ator. A Figura 5.17 mostra o resultado da eleição de um ator, para a tarefa ativa, no processo selecionado para ser executado. A tarefa ativa é assinalada na estrutura de dados **Processo**, registrando a última tarefa sendo executada, ou a próxima a ser executada. O roteamento é seqüencial e apenas uma instância do processo é executada de cada vez. Os dados exibidos na Figura 5.17 são comentados no capítulo 6.

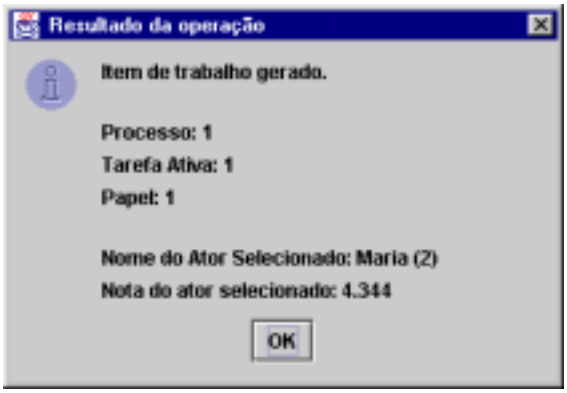

Figura 5.17 – Geração do Item de Trabalho

Quando há apenas um ator desempenhando o papel funcional para a tarefa, o sistema gera automaticamente um item para o mesmo e informa o resultado, conforme mostrado na Figura 5.18.

Se não houver ator com o papel funcional para a atividade, o sistema emite uma mensagem e não é gerado item de trabalho e nem invocada a eleição do ator para a atividade.

Figura 5.18 – Geração do Item de Trabalho sem Eleição do Ator

A execução do item de trabalho é realizada no WfFuzzy, a partir da exibição dos itens pendentes de cada instância dos processos em execução. O sistema permite escolher qual item executar, a partir de uma única lista de trabalho. A abstração, com relação às listas de trabalho em WfMS reais, ocorre na exibição dos nomes do Processo, da Tarefa e do Ator para cada item a ser executado, conforme mostrado na Figura 5.19, onde são mostrados os momentos da escolha do item e da sua execução. A simplificação, neste caso, envolve informar a **data** e a **hora** de início e fim de execução, além da **aceitação** ou **não** do resultado, pelo controle de qualidade. Após a confirmação da operação, o sistema atualiza a base de dados de itens de trabalho, recupera o identificador da próxima tarefa a executar e atualiza os dados de processo com a Tarefa Ativa. Se não houver uma próxima tarefa no processo, o sistema assinala a primeira tarefa do mesmo como ativa, considerando encerrada a execução da instância do processo.

O WfFuzzy executa apenas uma tarefa de cada vez, pois o que de visa demonstrar é a alocação de uma atividade. No caso de um WfMS real, as atividades em execução concorrente são um fator que diz respeito à sobrecarga do sistema. Se um ator está executando mais de uma atividade ao mesmo tempo, os estados das mesmas representarão a carga alocada ao mesmo, o que é coberto pelo trabalho.

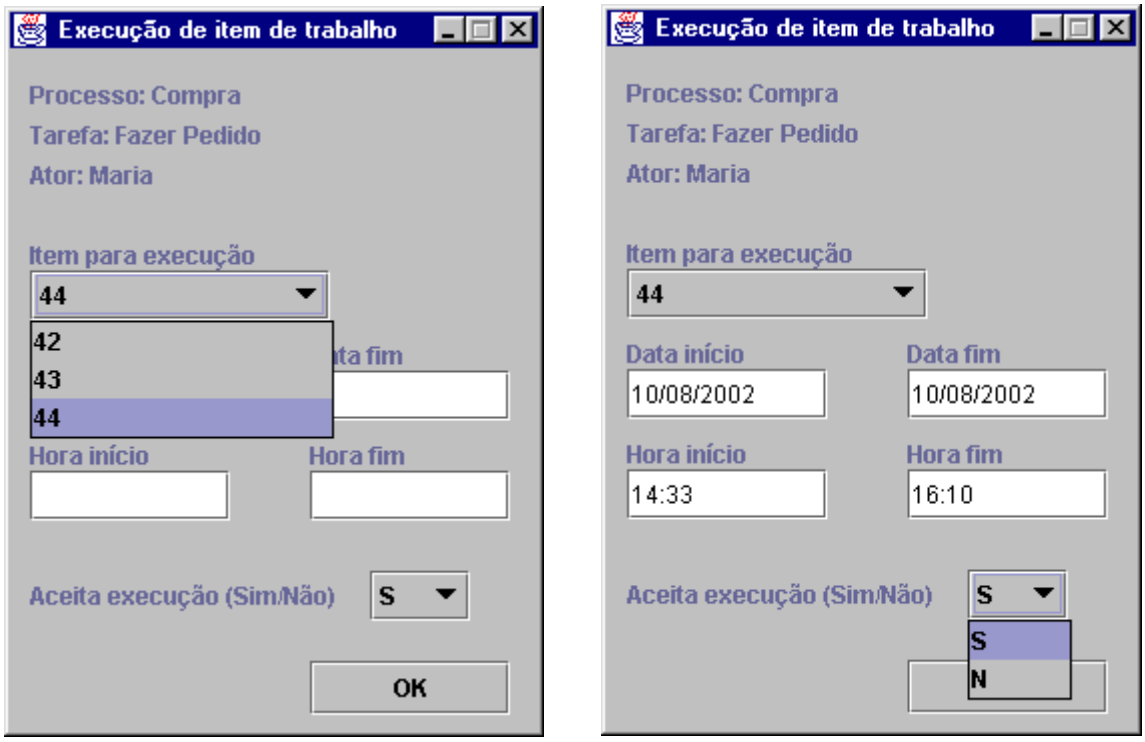

Figura 5.19 – Execução do Item de Trabalho

CAPÍTULO 6 - RESULTADOS OBTIDOS

Neste capítulo são discutidos os resultados da ação do WfFuzzy sobre definições de processos. São demonstrados os parâmetros para operação em uma simulação de processo em *workflow*. Os processos foram definidos de forma genérica, em situações simuladas, pois o objetivo é a comprovação das funcionalidades de eleição dos atores candidatos à execução das atividades.

Foram definidos três processos e realizadas 40 execuções de atividades, atribuindo itens de trabalho aos atores mais qualificados.

A Tabela 6.1, na página a seguir, traz as definições dos conjuntos nebulosos para a atividade Solicitar Compra, do processo Compra. A definição mostrada é a mesma adotada para todas as Atividades dos processos, usados para validar a ferramenta, pois o trabalho não se presta a analisar processos reais, mas comprovar, entre outras, a funcionalidade da proposta e sua flexibilidade em diferentes situações. Desta forma, na análise de eleições de atores, feita a seguir, são realizadas, ao final, alterações nos universos de discurso de variáveis específicas para uma atividade, a fim de verificar o impacto na mudança de comportamento do sistema.

Tabela 6. 1 – Configuração dos Conjuntos Nebulosos para cada Tarefa

Conjuntos Específicos		**Processo 1 – Compra**		**Tarefa 1 - Solicitar Compra**				
Variável				**Conjuntos**				
Nome	**Referencia**	**Tipo**	**Universo**	**Nome**	**Tipo**	**Ponto1**	**Ponto2**	**Ponto3**
Quantidade de Campos do Formulário	QtCamposFE	E	0 a 30	Mínima	TP	1	8	
				Baixa	TR	1	8	15
				Média	TR	8	15	22
				Alta	TR	15	22	29
				Máxima	TP	22	29	
Unidades de Tempo Externo	QtUnidTempo	E	0 a 150	Mínima	TP	5	40	
				Baixa	TR	5	40	75
				Média	TR	40	75	110
				Alta	TR	75	110	145
				Máxima	TP	110	145	
Grau de Dificuldade da Tarefa	GrauDificuldade	I	0 a 10	Mínimo	TP	1	3	
				Baixo	TR	1	3	5
				Médio	TR	3	5	7
				Alto	TR	5	7	9
				Máximo	TP	7	9	
Carga de Prioridade Alocada	CargaAlocada	E	0 a 20	Mínima	TP	1	5	
				Baixa	TR	1	5	10
				Média	TR	5	10	15
				Alta	TR	10	15	19
				Máxima	TP	15	19	
Prioridade da Tarefa	Prioridade	E	0 a 5	Máxima	TP	0	1	
				Alta	TR	0	1	2
				Média	TR	1	2	3
				Baixa	TR	2	3	4
				Mínima	TP	3	4	
Disponibilidade do Ator	Diponibilidade	I	0 a 10	Mínima	TP	1	3	
				Baixa	TR	1	3	5
				Média	TR	3	5	7
				Alta	TR	5	7	9
				Máxima	TP	7	9	
Antigüidade da Execução	AntigExecucao	E	0 a 60	Mínima	TP	3	16	
				Baixa	TR	3	16	30
				Média	TR	16	30	44
				Alta	TR	30	44	57
				Máxima	TP	44	57	
Desvio do Tempo Padrão	DesvioTExec	E	0 a 150	Mínimo	TP	10	40	
				Baixo	TR	10	40	70
				Médio	TR	40	70	100
				Alto	TR	70	100	130
				Máximo	TP	100	130	
Nota de Execução	NotaExec	I	A mesma definição de Disponibilidade do Ator					
Média de Execução	MediaExec	I	A mesma definição de Disponibilidade do Ator					
Execuções Rejeitadas	ExecRejeitadas	E	0 a 10	Mínima	TP	0	1	
				Baixa	TR	0	1	2
				Média	TR	1	2	4
				Alta	TR	2	4	7
				Máxima	TP	4	7	
Aproveitamento	Aproveitamento	I	A mesma definição de Grau de Dificuldade					
Rendimento	Rendimento	I	A mesma definição de Grau de Dificuldade					
Nota do Ator	NotaAtor	S	A mesma definição de Disponibilidade do Ator					

A definição de processos para a execução do WfFuzzy, foi feita em um estudo de caso genérico, e é apresentada nas Tabelas 6.2, 6.3 e 6.4. A cada atividade está associado o papel funcional exigido para desempenhá-la e os atores candidatos à sua execução.

Tabela 6. 2 – Papéis Funcionais no Ambiente do Sistema

Papel Funcional	
1	Apoio Administrativo
2	Analista de Compras
3	Chefe de Almoxarifado
2	Analista de Compras
4	Almoxarife
5	Aux. Financeiro
6	Chefe Financeiro
7	Caixa
8	Encarregado de Patrimônio

Tabela 6. 3 – Atores do Ambiente

Atores	
1	João
2	Maria
3	Pedro
4	Anelise
5	Elida
6	Evandro
7	Bruno
8	Celso
9	Alex
10	Rosa
11	Ricardo
12	Eloisa
13	Marcos

Na Tabela 6.4, o mesmo ator pode aparecer em mais de uma linha, relacionado à Tarefa, indicando que um ator pode ter mais de um papel funcional dentro da organização, ou mesmo dentro de um processo.

Tabela 6. 4 – Tarefas e Atores com os Papéis Funcionais nos Processos

Processo 1 - Compra			
Tarefa		**Papel**	**Atores**
1	Fazer Pedido	1	2, 8 e 10
2	Aprovar Pedido	2	1 e 5
3	Cotar Preços	1	2, 8 e 10
4	Analisar Cotação	2	1 e 5
5	Emitir OC	3	6
6	Receber Material	4	3
Processo 2 - Pagamento			
1	Enviar para Pagamento	1	8 e 9
2	Fazer Empenho	5	4 e 7
3	Aprovar pagamento	6	12
4	Pagar	7	13
5	Fazer Registro Contábil	1	8 e 9
Processo 3 - Alienação Patrimonial			
6	Pedir Alienação	1	2, 9 e 10
7	Aprovar Alienação	8	11
8	Alienar M P	1	2, 9 e 10

A seguir, é feita a amostragem dos resultados de execuções do WfFuzzy relativas a uma atividade, para comprovação dos resultados. As regras de inferência seguiram as configurações dos gráficos da seção 5.3. Os resultados são acompanhados de comentários, que visam contextualizar cada eleição de ator para a distribuição de tarefas, bem como cada execução de item de trabalho, como forma de comprovar o funcionamento do WfFuzzy. Os exemplos a seguir consideraram a **Tarefa 1** do **Processo 1**, identificados na Tabela 6.4 . A Tabela 6.5 exibe os dados da atividade, além das execuções, recuperadas do histórico do ator, necessários para a eleição. Dentre as execuções mostradas na tabela, são listados os itens de trabalho não executados pelo ator, diferenciados pela indicação da carga de trabalho alocada para cada item.

O primeiro cálculo realizado com lógica nebulosa é o do grau de dificuldade da tarefa, que resultou no valor 3,37 para os dados de entrada, considerados na Tabela 6.5. O Quadro E.1, do Anexo E, na página 139, exibe o trecho de código em Java para o

cálculo do Grau de Dificuldade da Tarefa e a gravação do valor na base de dados da Tarefa. No quadro E.2, do mesmo anexo, é mostrado o código da classe da variável GrauDif, que guarda as regras de inferência para o cálculo do valor do Grau de Dificuldade da Tarefa. As demais variáveis calculadas, apresentam codificação semelhante à da classe GrauDif. As classes que configuram as variáveis para a eleição do ator são herdadas da super classe Fuzzy, conforme exibido no diagrama de classes do sistema, na Figura C.1 do Anexo C, na página 135.

Tabela 6. 5 – Eleição do Ator para a Atividade – Configuração 1

Processo 1	**Tarefa 1**	**Data da distribuição: 10/08/02**	**Prioridade = 3**

EXECUÇÕES

Ator 2 – Maria

	Quant. De Campos	Unidade De Tempo Externo	Grau de Dificul-dade	Tempo Médio (h)	Prioridade do Item de Trabalho	Data de Distribuição e Execução.	Exec. Aceita	Duração (h)	Desvio de Exec. (%)	Antigui-dade de Execução (dias)	Nota de Execução
1	20	20	3,37	2	3	16/06/02	S	01:00	0(-50%)	65	5
2					3	22/07/02	S	01:30	25(-25%)	19	7,6419
3					3	07/08/02	S	01:00	0(-50%)	3	9,5
4					3	10/08/02	**Carga alocada = 2**				
5					2	10/08/02	**Carga alocada = 3**				
Ator 8 – Celso											
6	20	20	3,37	2	3	10/06/02	N	03:00	100(+50%)	75	5
7					3	16/07/02	S	03:30	125(+75%)	13	1,375
8					3	09/08/02	**Carga alocada = 2**		**Processo 3**		**Tarefa 3**
9					3	10/08/02	**Carga alocada = 2**				

ELEIÇÃO DO ATOR

	Carga Alocada	Disponibilidade	Média Das Execuções	Execuções. Rejeitadas (0/10)	Aproveitamento	Rendimento	Nota do Ator
Ator 2	5	5,27272	7,3806	0	7,47575	5,8874	**4,34**
Ator 8	4	5,72727	3,1875	1	2,67206	2,6913	**1,19465**

Na **amostra 7** é registrado um item de trabalho não executado, pertencente á Tarefa 3 do Processo 3.

- **Nota de Execução** – Na Tabela 6.5, pode ser observado que os desvios de execução da **amostra 1**, com tempo de execução na metade do tempo padrão, e da **amostra 6**, com tempo 50% acima do tempo padrão, resultaram

em nota de execução média, comprovando que o tempo, definido pela **Antigüidade de Execução**, ameniza resultados muito bons ou muito ruins. Por outro lado, a diferença entre as nota de execução das amostras 6 e 7 foi acentuada, apesar da diferença do desvio ser de apenas 25% entre as amostras. Isto é devido à pequena antigüidade, na **Amostra 7**, reforçando efeitos sobre a nota e a média das execuções do **Ator 8**, penalizando-as, neste caso.

- **Aproveitamento** – O fato do **Ator 2** não ter execuções rejeitadas fez com que seu aproveitamento se mantivesse em alta pontuação, em relação à sua média de execuções. O mesmo não ocorreu para o **Ator 8**.

- **Rendimento** – O rendimento do **Ator 2** caiu, em relação ao seu aproveitamento, pois o grau de dificuldade da tarefa a distribuir é baixo, não requerendo atores de grande aproveitamento. No caso do **ator 8**, o sistema atribuiu valor de rendimento mais alto que o do aproveitamento, mostrando que, para tarefas com baixo grau de dificuldade, atores de baixo aproveitamento podem ser escolhidos, tendo a chance de melhorar sua atuação.

- **Nota do Ator** – Devido à baixa diferença de disponibilidade entre os atores **2** e **8**, as diferenças entre suas notas finais mantiveram em uma relação de linearidade, em relação aos seus rendimentos. Desta feita, o sistema WfFuzzy comunica o resultado da eleição para o motor do WfMS que, através do componente Gerenciador da Lista de Trabalho, gera um item de trabalho para o ator 2.

O resultado da eleição do ator pelo WfFuzzy é exibida na Figura 6.1.

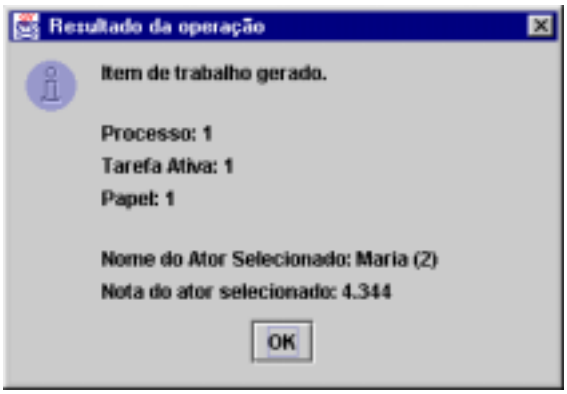

Figura 6. 1 – Resultado da Eleição pelo WfFuzzy

A seguir, na Tabela 6.6, são exibidos os dados de outro contexto de execuções, seguidos de comentários sobre os resultados.

Tabela 6. 6 – Eleição do Ator com Alterações de Parâmetros (1)

Processo 1	**Tarefa 1**	**Data da distribuição: 10/08/02**	**Prioridade = 3**

EXECUÇÕES

Ator 2 – Maria

	Quant. De Campos	Unidade De Tempo Externo	Grau de Dificul-dade	Tempo Médio (h)	Priori-dade do Item de Trabalho	Data de Distribuição e Execução.	Aceita	Duração (h)	Desvio de Exec. (%)	Antigui-dade de Execução (dias)	Nota de Execução
1	20	20	3,37	2	3	16/06/02	S	01:00	0(-50%)	65	5
2					3	22/07/02	**N**	01:30	25(-25%)	19	7,6419
3					3	07/08/02	**N**	01:00	0(-50%)	3	9,5
4					3	10/08/02	**Carga alocada = 2**				
5					2	10/08/02	**Carga alocada = 3**				
Ator 8 – Celso											
6	20	20	3,37	2	3	10/06/02	N	03:00	100(+50%)	75	5
7					3	16/07/02	S	03:30	125(+75%)	13	1,375
8					3	09/08/02	**S**	**01:30**	**25(-25%)**	**1**	**9,5**
9					3	10/08/02	**S**	**01:00**	**0(-50%)**	**0**	**9,5**

ELEIÇÃO DO ATOR

	Carga Alocada	Disponibilidade	Média Das Execuções	Execuções. Rejeitadas (0/10)	Aproveitamento	Rendimento	Nota do Ator
Ator 2	5	5,27272	7,3806	**2**	**5,3806**	**5,27**	**3,85253**
Ator 8	**0**	**7**	**6,3475**	1	**5,6036**	**5,27**	**5,27**

A Tabela 6.6 ilustra alterações nas variáveis de entrada, como forma de verificar o comportamento inteligente do sistema, em contraste com os resultados originais da Tabela 6.5.

- Ao serem registradas duas **execuções rejeitadas** para o **ator 2**, houve um reflexo em seu aproveitamento (5,27). Como o novo aproveitamento situa-se na média da escala do seu universo de discurso, o rendimento não sofreu alterações significativas, pois as regras de inferência do sistema atribuem valor médio de rendimento para graus de dificuldade baixos. A **nota** do ator 2 foi alterada, de 4,34 para 3,85253, em função do seu novo rendimento. Não foi uma alteração muito grande, pois o ator acusa altas notas de execução para suas tarefas, e sua disponibilidade não foi alterada.

- O efeito sobre a disponibilidade pode ser notado ao serem registradas as **execuções** de dois itens pendentes para o **ator 8**. Como as execuções foram realizadas abaixo do tempo padrão, elas acusaram altas notas de execução, o que aumentou a média de execuções do ator. Todavia, o efeito de uma execução ruim, com nota igual a 1,3575 ainda manteve a média baixa em relação às altas notas recentes. Com isso, o sistema amplia a pontuação do ator, mas leva em consideração efeitos de outras execuções, equilibrando variações muito grandes de desempenho, que podem indicar inconstância de comportamento.

- A execução rejeitada que o **ator 8** acusa em seu histórico, ainda penaliza seu aproveitamento, mas em relação menor que a registrada na Tabela 6.5, dado o acréscimo em sua média de execuções. Desta forma, o sistema mostra a recuperação do aproveitamento do ator, em função de seus bons resultados.

- O rendimento do **ator 8** apresentou valor médio, em decorrência da situação explicada para o **ator 2** na Tabela 6.5.

- A nota final do **ator 8** subiu de forma considerável, mostrando que, para o contexto da Tabela 6.5, relativo ao **ator 2**, ele seria o eleito.

A Tabela 6.7 mostra que uma **alteração no universo de discurso** da variável Antigüidade de Execução muda o comportamento do sistema.

Tabela 6. 7 – Eleição do Ator com Alterações de Universo de Discurso (2)

Processo 1		**Tarefa 1**			**Data da distribuição: 10/08/02**					**Prioridade = 3**
EXECUÇÕES										
Ator 2 – Maria										
	Quant. De Campos	Unidade De Tempo Externo	Grau de Dificul-dade	Tempo Médio (h)	Prioridade do Item de Trabalho	Data de Distribuição e Execução.	Aceita	Duração (h)	Desvio de Exec. (%)	Antigui-dade de Execução (dias)	Nota de Execução
1	20	20	3,37	2	3	16/06/02	S	01:00	0(-50%)	65	**8,54321**
2					3	22/07/02	N	01:30	25(-25%)	19	**9,08678**
3					3	07/08/02	N	01:00	0(-50%)	3	9,5
4					3	10/08/02	**Carga alocada = 2**				
5					2	10/08/02	**Carga alocada = 3**				
Ator 8 – Celso											
6	20	20	3,37	2	3	10/06/02	N	03:00	100(+50%)	75	5
7					3	16/07/02	S	03:30	125(+75%)	13	**1,21907**
8					3	09/08/02	S	01:30	25(-25%)	1	9,5
9					3	10/08/02	S	01:00	0(-50%)	0	9,5

ELEIÇÃO DO ATOR

	Carga Alocada	Disponibilidade	Média Das Execuções	Execuções. Rejeitadas (0/10)	Aproveitamento	Rendimento	Nota do Ator
Ator 2	5	5,27272	**9,04333**	2	**7**	**5,37**	**3,93353**
Ator 8	0	7	**6,30477**	1	**5,57695**	5,27	5,27

A alteração do universo de discurso da variável Antigüidade de Execução, de **0 a 60**, para **0 a 180**, fez com que os valores de nota de execução do **ator 2** subissem muito, pois o sistema passou a considerar recentes alguns resultados que já estavam na média, devido à obsolescência causada pela grande antigüidade da execução, no contexto da Tabela 6.6. Os valores resultantes desta alteração, são resultados de comportamentos já discutidos nos dois contextos anteriores de execução. Com os resultados da Tabela 6.7, ficou claro que o sistema apresenta flexibilidade e capacidade de adaptação a uma situação de contingência, relativa a determinadas atividades, com alterações nos parâmetros de definição do processo, demonstrando **comportamento inteligente**. Desta forma, uma política pode ser aplicada de maneira geral para todos os atores em um processo. Uma tarefa pode ter características alteradas, sem afetar o comportamento das outras no processo, no tocante à eleição de atores.

CAPÍTULO 7 - CONSIDERAÇÕES FINAIS

7.1 CONCLUSÕES

Este trabalho propôs um mecanismo chamado WfFuzzy, para atribuição de notas a atores, em processos controlados por sistemas de *workflow*. A proposta é baseada em lógica nebulosa e realiza inferência sobre dados de definição de processos e de histórico de execução de atividades dos atores candidatos às atividades.

Pôde ser observado que o sistema apresentou comportamento inteligente, em decorrência das alterações nas variáveis nebulosas de entrada, escolhendo o ator mais qualificado para o desempenho de uma atividade. As amostras analisadas no capítulo 6 mostraram que o sistema apresenta alterações de resultado na escolha do melhor ator, compatíveis com o comportamento de um gerente humano, que fizesse a escolha.

A configuração das variáveis nebulosas mostrou que o sistema é adaptável para diferentes situações. Todavia, como esta configuração deve ser feita para cada atividade no processo, a implantação e manutenção das variáveis exige um volume de trabalho muito grande para uma situação real. A definição automática das funções de pertinência pode facilitar a modelagem do processo e é comentada com maiores detalhes na seção 7.3.

Pôde-se constatar que alterações nos universos de discurso podem adaptar as variáveis para diferentes instalações de um sistema de *workflow* distribuído, que apresente política homogênea para um determinado processo, com diferenças apenas quantitativas entre as realidades do negócio.

Pode-se concluir que o WfFuzzy se constitui em um mecanismo no qual o executor determina os parâmetros de configuração dinâmica de distribuição do trabalho, através da geração de seu histórico de comportamento, sendo agente e reagente neste processo, onde os efeitos de ganho ou perda de produtividade pessoal são sentidos pelo grupo e têm que ser por ele resolvidos.

A aplicação da solução proposta pode sobrecarregar o ator eficiente, que apresenta melhores resultados. Deve-se considerar o papel da gerência de pessoal, incentivando, e corrigindo distorções.

A extensão deste trabalho para permitir a definição dinâmica das regras de inferência ampliaria a aplicação do WfFuzzy, tornando-o adaptável a situações onde os contextos de negócio determinam diferentes abordagens para um mesmo problema.

A aplicação do sistema WfFuzzy em um WfMS real, exige a adoção de políticas de acesso e manutenção dos dados relevantes de *workflow* que, para o nível de sofisticação dos melhores sistemas do mercado, tornaria a solução proposta aplicável, desde que desenvolvida especialmente para interação com o motor do sistema. Isto se deve ao fato do motor congregar as funcionalidades que diferenciam os sistemas e, portanto, não possuir arquitetura disponível para acesso a terceiros.

7.2 DIFICULDADES ENCONTRADAS

Durante a realização do trabalho alguns aspectos se destacaram como dificuldades para o andamento dos trabalhos.

A primeira dificuldade encontrada foi relativa à determinação da abrangência do problema considerado. Isto ocorreu, em maior grau, devido ao pouco conhecimento disponível em sistemas de *workflow*, para determinar um problema real no desempenho destes sistemas. O tema foi pesquisado na Internet, em livros técnicos, e na troca de experiências com profissionais da área. Pôde-se constatar a dificuldade no entendimento da tecnologia. Não há uma definição específica de estrutura de um WfMS do tipo produção, que permita entender de maneira mais direta o funcionamento de um sistema desta categoria. A solução para isso foi buscar informações sobre órgãos de padronização da tecnologia, o que levou ao estudo das especificações da WfMC. As informações sobre o funcionamento de um WfMS, em um ambiente real de negócio, forma obtidas através de contato direto com o prof. Tadeu Roberto Cruz, da TrCr e-Process, de São Paulo.

A abordagem exclusivamente técnica para um WfMS, foi tentada na primeira fase dos estudos, com investigações sobre balanceamento de carga na execução destes sistemas. A falta de informações, por parte das empresas que desenvolvem sistemas de *workflow*, e a escassez de bibliografia voltada a estes pontos, levou à determinação de uma abordagem que enfocasse o aspecto humano de rendimento, em processos executados por WfMS.

No tocante à simulação do comportamento inteligente, com ferramentas de modelagem nebulosa, a dificuldade inicial foi a visualização dos efeitos das variáveis de entrada no resultado final, considerando o número de variáveis intermediárias. Este problema foi solucionado com o uso do *software* FuzzyTECH, comentado no capítulo 5. Este aplicativo permite que sejam feitas simulações, apresentando resultados gráficos interativos, permitindo a visão dos resultados em diversos pontos do modelo sendo desenvolvido.

7.3 TRABALHOS FUTUROS

O objetivo proposto por este trabalho foi atingido, na comprovação do comportamento inteligente de uma aplicação que apoie sistemas de *workflow*. Todavia, o sistema WfFuzzy pode ser sofisticado para simulação mais real de um WfMS e há trabalhos a serem desenvolvidos, que possam completar a idéia inicial, configurada nesta proposta. Dentre as perspectivas futuras para a continuidade das pesquisas feitas até então, pode-se destacar:

- O uso de tecnologias XML, para homogeneizar bases de dados heterogêneas. Uma das tecnologias a serem pesquisadas é a Linguagem de Definição de Processos em XML (XML *Process Definition Language*), lançada pela WfMC [Wfmc02]. Com uma definição que permita intercâmbio entre sistemas de *workflow*, a aplicação da solução proposta neste trabalho é facilitada.

- A determinação dinâmica das regras de inferência, tornando a solução aplicável a um universo mais extenso de problemas. Isto pode ser feito definindo-se uma estrutura de dados de regras, a serem combinadas de acordo com as necessidades específicas de cada situação. Neste caso, uma rede neural pode ser usada para

definição da base de regras no sistema nebuloso, compondo uma solução híbrida de IA para sistemas de *workflow*.

- O uso de agentes de *software*, possuindo mobilidade, em sistemas de *workflow* distribuídos, transportando a aplicação WfFuzzy, e os resultados da eleição do melhor ator em cada instalação visitada, pode contribuir para o incremento da eficiência global do sistema, na medida em que tarefas podem requerer a intervenção de atores pertencentes a sistemas geograficamente distribuídos. Os agentes podem transportar a aplicação até onde se encontram os dados a serem analisados, reduzindo o tráfego na rede e trabalhando com dados processados em tempo real. Neste caso, deve-se considerar a estratégia de distribuição dos agentes, de modo evitar que eles encontrem dados desatualizados, ao completar sua tarefa, em decorrência do tempo gasto na eleição e no deslocamento. Dentre outras funções que podem ser desempenhadas por este tipo de agente inteligente de *software*, estão a verificação da execução das tarefas distribuídas, a emissão de avisos, ou a tomada de decisão para redistribuição de tarefas, de acordo com o desempenho dos atores.

- O uso de variáveis aleatórias para avaliação do desempenho dos atores. A abordagem dada a este trabalho, considerou variáveis fixas, utilizadas para classificação, parametrizadas por especialistas em avaliação de desempenho humano, em processos de negócio. Sistemas de *workflow* atuais dependem da interação fluente e eficiente entre atores humanos e sistemas artificiais, apresentando uma crescente complexidade em se automatizar o planejamento e a execução de atividades [Hann99]. A determinação das variáveis poderia ser feita a partir da análise do modelo de *workflow*, específico para cada instalação, ou tipo de processo em execução. Um desafio, neste caso, seria parametrizar o tipo de variável e o resultado esperado para análise da mesma, ou de grupos de variáveis que influenciam determinado resultado. Esta parametrização poderia compor a formação das regras de avaliação para o mecanismo de inferência., otimizando a associação de atividades a atores, flexibilizando a aplicação proposta.

- A geração das funções de pertinência, no sistema nebuloso, a partir de dados reais, através de uma solução híbrida, com rede neural e lógica nebulosa. A utilização de

treinamento, com propagação recursiva de erros, permite que sejam encontrados valores para as funções de pertinência, de forma a exigir menor conhecimento dos paradigmas de IA, por parte dos administradores do sistema, onde a solução se aplica.

REFERÊNCIAS BIBLIOGRÁFIAS

[Alle01] ALLEN, R., 2001, *Workflow: Na Introduction – The Workflow Handbook 2001*, Future Strategies Inc., Florida.

[Alve00] ALVES , A. C. F. C., 2000, Definição de um Modelo de *Workflow* Integrando a Cooperação e a Organização Temporal, Dissertação de Mestrado, Universidade Federal do Maranhão, São Luis, MA.

[Atlu01] ATLURI, V., 2001, *Security for Workflow Systems*, *Information Security Technical Report*, v 06 pp.59-68.

[Barr00] BARRETO, J. M., 2000, Inteligência Artificial no Limiar do Século XXI, Duplic – Prestação de Serviços, Florianópolis.

[Bene00] BENEDICT, K., 2000, *Workflow Enables Companies to Extract Efficiency from Processes*, *In: Workflow Feature, Computerweek*, Disponível em www.computerweek.co.za/ecwk/2000/001002/articles/*workflow*.htm, Acesso em: 20 de Outubro de 2000.

[Bigu98] BIGUS, J. P., BIGUS, J., 1998, *Constructing Intelligent Agents with Java*, Wiley Computer Publishing, New York.

[Cruz98] CRUZ, T., 1998, *Workflow*: A Tecnologia que vai Revolucionar Processos, Atlas, São Paulo, SP.

[Cruz01] CRUZ, T., 2001, *E-Workflow*, CENADEM, São Paulo, SP.

[Eber96] EBERHART, R., SIMPSON, P., DOBBINS, R., 1996, *Computational Inteligence PC Tools*, AP Professional, London.

[Elma00] ELMASRI, R.., NAVATHE, B. S., 2000, *Fundamentals of Database Systems*, 3.ed., Addison-Wesley, USA.

[Fann01] FANNING, B., 2001, *The Value of Standards*, *The Workflow Handbook* 2001, Future Strategies Inc., Florida.

[Galv99] GALVÃO, C. º, VALENÇA, M. J. S., VIEIRA, V. P. P. B., DINIZ, L. S., LACERDA, E. G. M., CARVALHO, A., LUDERMIR, T. B., 1999, Sistemas Inteligentes: Aplicações a Recursos Hídricos e Ciências Ambientais, ABRH/UFRGS, Editora da Universidade.

[Grae97] GRAEBER, S., 1997, *The Impact of Workflow Management Systems on the Design of Hospital Information Systems*. University of Saarland, Germany, Disponível em: www.amia.org/pubs/symposia/DOO4101.PDF, Acesso em: 15/08/2001.

[Hage99] HAGEN, C. J., 1999, *A Generic Kernel for Reliable Process Support*, Tese de PhD, Swiss Federal Institute of Technology, Zürich, Switzerland.

[Hags00] HAGSTRÖM, A., FAK, V., 2000, EWS – *A Case Study on Access Control in Workflow Systems*, *In: Proceedings of the IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Colaborative Enterprises*.

[Hann99] HANNEBAUER, M., 1999, *From Formal Workflow Models to Intelligent Agents*, GMD – German National Center for Information Technology.

[Hayk01] HAYKIN, S., 2001, Redes Neurais – Princípios e Prática, Bookman, Porto Alegre.

[Holl95] HOLLINGSWORTH, D., 1995, *Workflow Management Coalition: the Workflow Reference Model*, Document Number TC00-1003, Hampshire, UK.

[Holl01] HOLLINGSWORTH, D., 2001, *The WfMC Glossary*, *Workflow Handbook 2001*, Future Strategies Inc. , Florida/USA.

[Klir95] KLIR, G. J., YUAN, B., 1995, *Fuzzy Sets and Fuzzy Logic – Theory and Applications*, Prentice Hall, New Jersey.

[Larm98] LARMAN, C., 1998, *Applying UML and Patterns*, Prentice Hall PTR, New Jersey.

[Leym00] LEYMANN, F., ROLLER, D., 2000, *Production Workflow – Concepts and Techniques*, Prentice Hall PTR, New Jersey.

[Mano01] MANOLESCU, D., 2001, *Micro-Workflow: A Workflow Arquitecture Supporting Compositional Object-Oriented Software Development*, Tese de PhD., Illinois University at Urbana Champaign.

[Merz97] MERZ, M., LIBERMAN, B., LAMERSDORF, W., 1997, *Using Mobile Agents to Support Interorganizational Workflow Management*, International Journal of Applied Artificial Intelligence, 11(6) pp. 551-572.

[Omg98] OMG, 1998, *Workflow Management Facility, Revised Submission*, *OMG Document Number*: bom/98/06/07, *July* 1998, Disponível em: http://cgi.omg.org/issues/issue2066.txt, Acesso em: 22/12/2000.

[Orfa97] ORFALI, R., HARKEY, D., EDWARDS, L., 1997, *Instant Corba*, Wiley Computer Publishing, Canadá.

[Schm99] SCHMIDT, M. T., 1999, *The Evolution of Workflow Standards*, IEEE Concurrency, September, pp. 44-52.

[Shar00] SHARP, A., McDERMOTT, P., 2000, *Workflow Modeling – Tools for Process Improvement and Application Development*, Artech House, Norwood.

[Shaw99] SHAW, I. S., SIMÕES, M. G., 1999, Controle e Modelagem Fuzzzy, Edgard Blücher Ltda.

[Shet99] SHETH, A. P., 1999, *Process Driving the Networked Economy*, IEEE Concurrency, July-Sept.

[Silv01] SILVA, F., SILVEIRA, R, *et al.*, 2001, *CORBA Based Architecture for Large Scale Workflow*, Instituto de Computação da Universidade de Campinas –

UNICAMP, Disponível em: www.computer.org/proceedings/isads/0137/01370276abs.htm, Acesso em 15 de fevereiro de 2001.

[Wfmc99] WfMC, 1999, *Workflow Security Considerations – White Paper*, Workflow Management Coalition, Document Number WfMC-TC-1019.

[Wfmc02] WfMC, 2002, *Workflow Process Definition Interface – XML Process Definition Language*, Workflow Management Coalition, Document Number WfMC-TC-1025.

[Zade73] ZADEH, L. A., 1973, *Outline of a New Approach in the Analysis of Complex Systems and Decision Processes*, IEEE Trans. On Systems, Man, and Cybernetics, v. 3, pp. 28-44.

GLOSSÁRIO

Ad Hoc **-** Feito, ou programado, somente quando a situação assim o fizer necessário, e sem planejamento prévio.

Agente – Programa que executa tarefas pequenas e bem definidas em *background*, ou seja, sem a interação com o usuário, geralmente acessando e alterando informações persistentes no sistema.

API – *Aplication Program Interface* – Grupo de *interfaces* para o uso de rotinas, protocolos e ferramentas que um sistema operacional, ou linguagem de programação, fornecem para a construção de um *software*, sem que se tenha que conhecer detalhes de implementação.

BPR – *Business Process Reengineering* – Conjunto de ações desempenhadas para a descoberta de processos de negócio e desempenhadas em engenharia de processos.

Componente – Um pequeno objeto binário, ou programa, que executa uma função específica e é projetado de forma a atuar de forma facilitada com outros componentes e aplicações.

CORBA – *Common Object Request Broker Architecture* – Uma arquitetura que permite que partes de programas, chamados objetos, possam se comunicar entre si, independente da linguagem em que foram escritos ou do sistema operacional onde estão sendo executados. CORBA foi desenvolvido por um consórcio de indústrias chamado OMG – *Object Management Group*.

CSCW – *Computer-Supported Cooperative Work* – Trabalho cooperativo apoiado por computador. Grupo de sistemas baseados em computadores, que suportam grupos de pessoas envolvidas em uma tarefa comum e fornecem *interface* para um ambiente compartilhado.

Data Mining – Uma classe de aplicações de bases de dados, que visam a descoberta de padrões ocultos em um grupo de dados, que podem ser usados para predizer um comportamento futuro. Isto é feito pela descoberta prévia de relacionamentos desconhecidos entre os dados.

Data Warehouse – Uma coleção de dados projetada para o gerenciamento da tomada de decisão. Contém uma variedade de dados que apresentam um quadro coerente das condições de negócio em um ponto singular no tempo.

Documento Eletrônico – Formato digital de um documento, necessário para sua utilização em um processo controlado por um sistema de *workflow*. O documento pode ser gerado a partir de um aplicativo computacional, ou ser digitalizado em um *scanner*.

ERP – *Enterprise Resource Planning* – Um sistema para gerenciamento de negócio, que integra todas os aspectos do mesmo, desde planejamento, manufatura, vendas e *marketing.*

Framework – Uma estrutura extensível, para descrever um grupo de conceitos, métodos, tecnologias e mudanças culturais necessárias para o projeto completo de um produto e um processo de construção. *Frameworks* provêm um mecanismo que dirige o usuário através de uma ordem apropriada de passos, aplicações, e conversões de dados, via uma interface comum, para o processo sendo seguido.

Groupware – Qualquer ferramenta que auxilie um grupo de pessoas a trabalharem juntas, de forma mais fácil e eficaz, procurando atender aos aspectos de comunicação de informações, coordenação de papéis pessoais e colaboração entre pessoas para cumprir um trabalho conjunto.

Inferência – Racionalização, através da qual conclusões são derivadas de premissas. Em IA, uma premissa pode ser tanto um fato quanto uma regra.

Instância – A representação de uma simples execução de um processo, ou de uma atividade em um processo, incluindo seus dados associados. Cada instância representa uma unidade de execução separada, do processo ou da atividade, podendo ser

controlada independentemente, possuindo seu próprio estado interno e sua identidade externamente visível, para fins de recuperação de dados de auditoria, entre outros.

Interoperabilidade – Habilidade que *software* e *hardware*, em diferentes máquinas, de diferentes fabricantes, tem, para compartilhar dados.

Linguagem de Definição de *Workflow* – Linguagem que reflete a estrutura básica de um meta modelo, e que permite a troca de definições de *workflow*, como modelos de processo, ou estruturas organizacionais, entre ferramentas de engenharia de negócio e um WfMS. AS construções da linguagem, como um modelo de processo, são identificadas por palavras-chave e recebem um nome único. As propriedades destas construções são identificadas por grupos de parâmetros, que por sua vez são identificados por cadeias de caracteres. Estas definições podem ser manipuladas por editores de texto disponíveis na plataforma de *software* utilizada para desenvolvimento.

Message Broker – Dispositivo em um sistema de enfileiramento de mensagens, que cuida da heterogeneidade da rede, onde trafegam aplicações para diferentes formatos e diferentes máquinas. Ele é responsável pela tradução de mensagens, pelo roteamento inteligente e processamento das regras contidas nas mensagens.

Meta Modelo – Estrutura genérica, para a construção de modelos de um ambiente, através da definição de um grupo básico de propriedades dos elementos característicos da realidade sendo modelada, e que permite a adição de novas propriedades.

MQI (*Message Queueing Invocation*) – Mecanismo de invocação de programas, no qual um componente é invocado através do envio de determinada mensagem a ele. O componente invocado lê a mensagem, executa as ações apropriadas e manda de volta outra mensagem que contém os dados resultantes, e outras informações como um código de retorno. Este mecanismo trabalha de modo assíncrono, ou seja, o objeto chamador envia a invocação e continua a trabalhar e, quando o componente envia a mensagem com o resultado de seu trabalho, ela é capturada pelo chamador, que continua com o seu trabalho.

Middleware – *Software* que facilita a comunicação entre sistemas de *software*. Provê uma API, através da qual as aplicações invocam serviços e controla a transmissão da troca de dados na rede.

Motor de Inferência – Componente de um sistema de lógica nebulosa, que opera em uma série de regras de produção e realiza inferências nebulosas.

Objeto de Negócio – Objetos reutilizáveis, ou componentes de servidor, que encapsulam lógica de negócio.

OMG – *Object Management Group* – Consórcio formado pela participação de centenas de companhias, com a meta de prover um *framework* comum para o desenvolvimento de aplicações, usando técnicas de programação orientada a objeto.

ORB – *Object Request Broker* – Componente de *software* da arquitetura CORBA, que age como intermediário em uma requisição a um objeto distribuído, feita por um cliente, em tempo de execução, liberando-o do conhecimento da forma de implementação deste objeto e de sua localização no servidor de objetos.

PDL – *Process Definition Language* – Linguagem para a especificação de processos, que descreve as tarefas individuais e suas relações de precedência. O padrão da WfMC para esta linguagem, *Workflow Process Definition Language* (WPDL), estabelece regras para a troca de definição de processos, utilizando objetos e atributos definidos no meta modelo de processos.

Função de Pertinência – Em um conjunto nebuloso, é o mapeamento de um conjunto de objetos, para o intervalo de 0 a 1.

Peso Sináptico – A força de uma conexão entre dois elementos de processamento, que determina o efeito em rede de um elemento de processamento sobre o outro. Conexões podem ter valor positivo, zero, ou negativo.

Program Call – Principal mecanismo para a execução de programas em uma estação de trabalho. É um mecanismo síncrono para a invocação de programas, fornecido pelo sistema operacional sendo utilizado, ou seja, o processo que invoca o programa, na estação de trabalho, aguarda até que ele seja executado.

RPC - *Remote Procedure Call* – Tipo de protocolo que um programa em um computador execute um outro programa em um computador servidor. Desta forma, o desenvolvedor que usa RPC não precisa desenvolver procedimentos específicos para o servidor. O programa **cliente** envia uma mensagem ao **servidor**, com os argumentos apropriados, e este retorna uma mensagem contendo os resultados do programa executado.

SGBD distribuído – Sistema gerenciador de banco de dados, que controla bases de dados espalhadas, ou distribuídas, por diversos sistemas de computador.

SQL – *Structure Query Language* – Linguagem Estruturada de Pesquisa, padronizada, para requisições de informações de uma base de dados.

UML – *Unified Modeling Language* – Linguagem Unificada de Modelagem, para especificar e visualizar projetos grandes e complexos de *software* orientado a objeto. A UML implementa métodos consagrados de desenvolvimento e é desenvolvida sob as definições da OMG.

Unidade de Execução – Uma parte de um programa, que pode ser executado independente de outras partes, de forma concorrente.

WAPI – *Workflow APIs and Interchange Formats* – Especificação para API, da WfMC, que incorpora especificações para habilitar interoperabilidade entre diferentes componentes de WfMS e aplicações.

WARIA – *Workflow And Reengineering International Association* – Entidade criada para identificar e esclarecer pontos comuns a usuários de *workflow*, comércio eletrônico e aos envolvidos na reengenharia de suas organizações. A WARIA trabalha em consonância com a WfMC.

XML – *Extensible Markup Language* – Uma especificação para a criação de comandos em documentos, que definem como eles devem ser formatados, com aplicação especialmente voltada a documentos para a Internet. Estes comandos são personalizados, permitindo a definição, transmissão, validação e interpretação de dados entre aplicações e organizações.

ANEXOS

ANEXO A – MODELO DE DADOS DO WfFuzzy

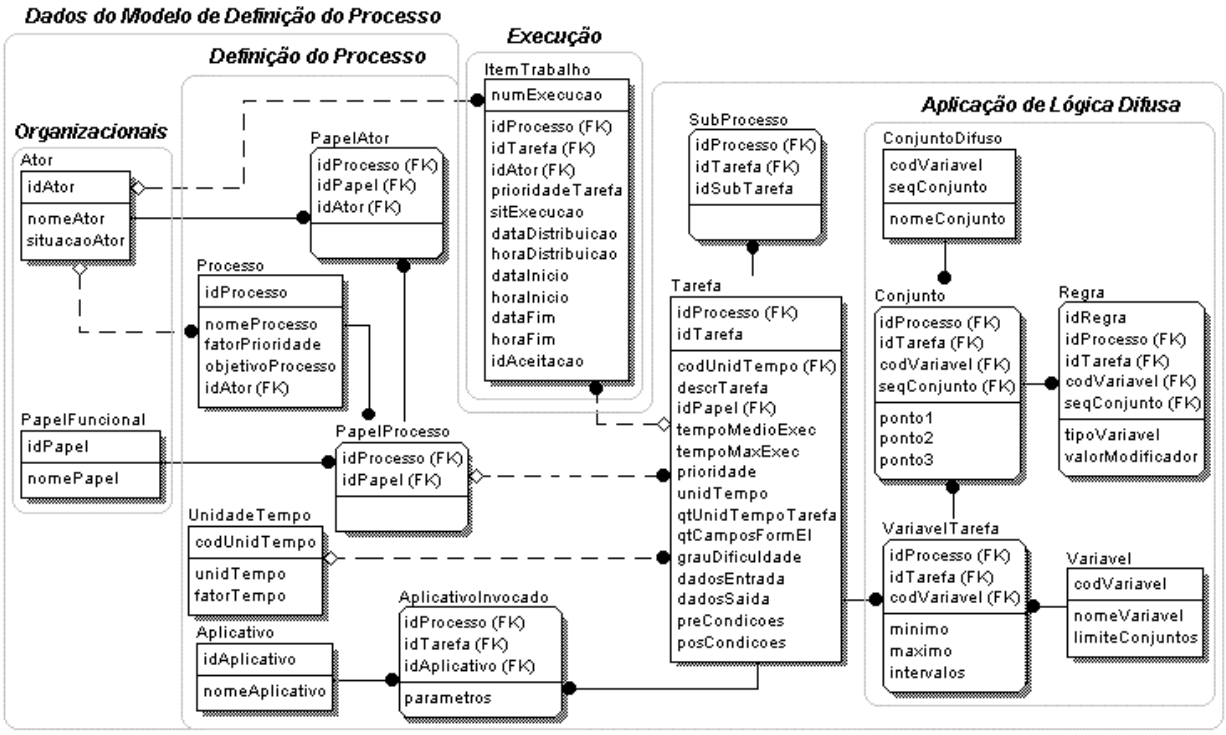

Figura A.1 – Modelo de Dados do WfFuzzy

ANEXO B – DIAGRAMAS DE CASOS DE USO E DE SEQÜÊNCIA

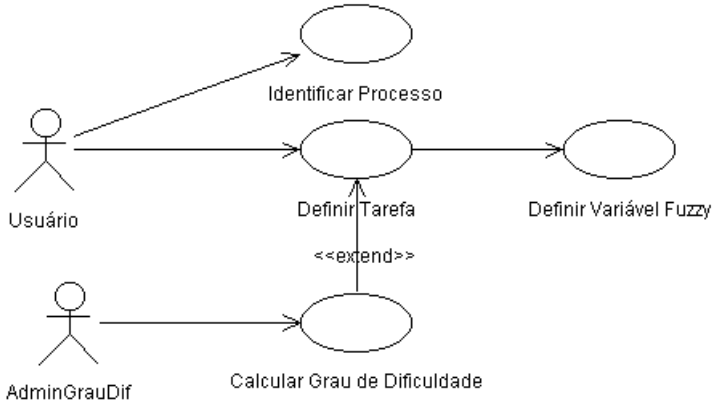

Figura B.1 – Diagrama de Caso de Uso - Definir Processo

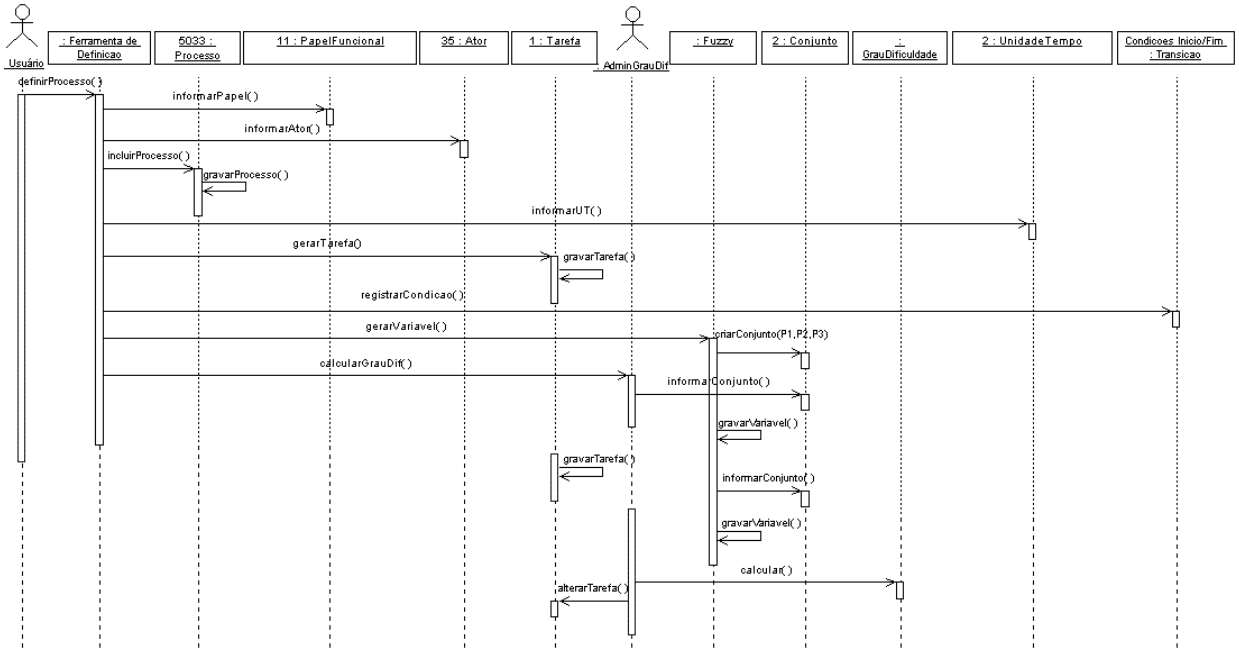

Figura B.2 – Diagrama de Seqüência – Definir Processo

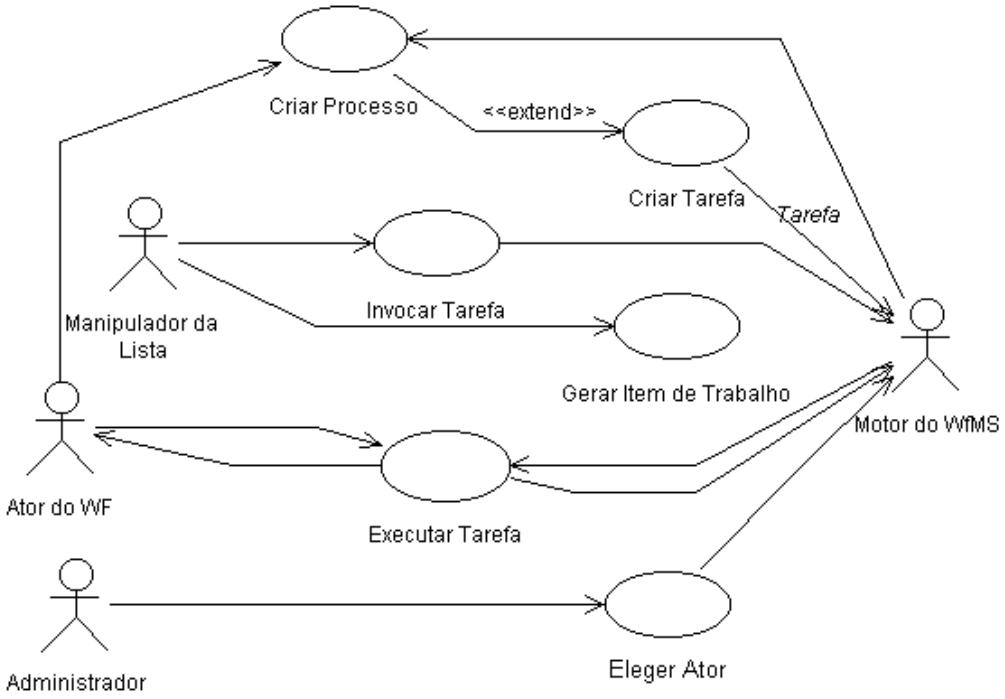

Figura B.3 – Diagrama de Caso de Uso – Executar Processo

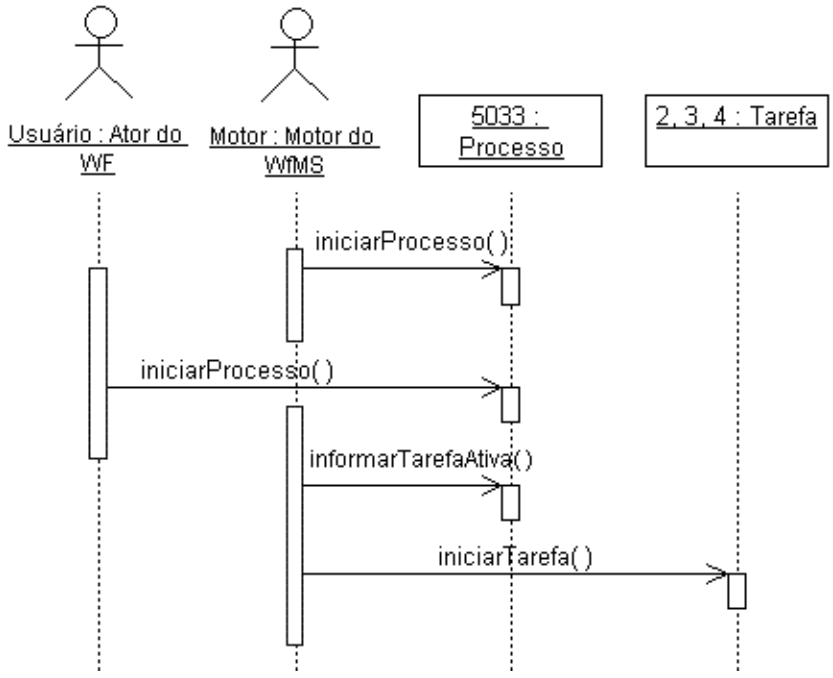

Figura B.4 – Diagrama de Seqüência – Criar Processo

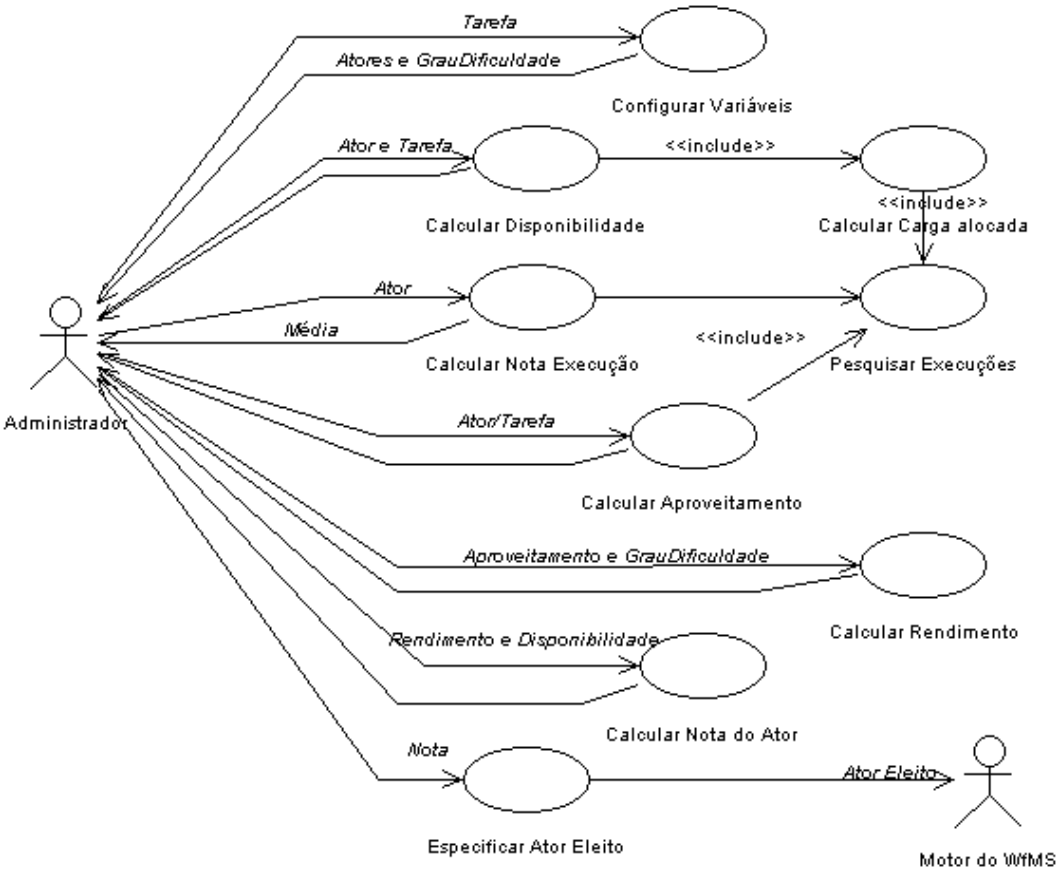

Figura B.5 – Diagrama de Caso de Uso – Eleger Ator

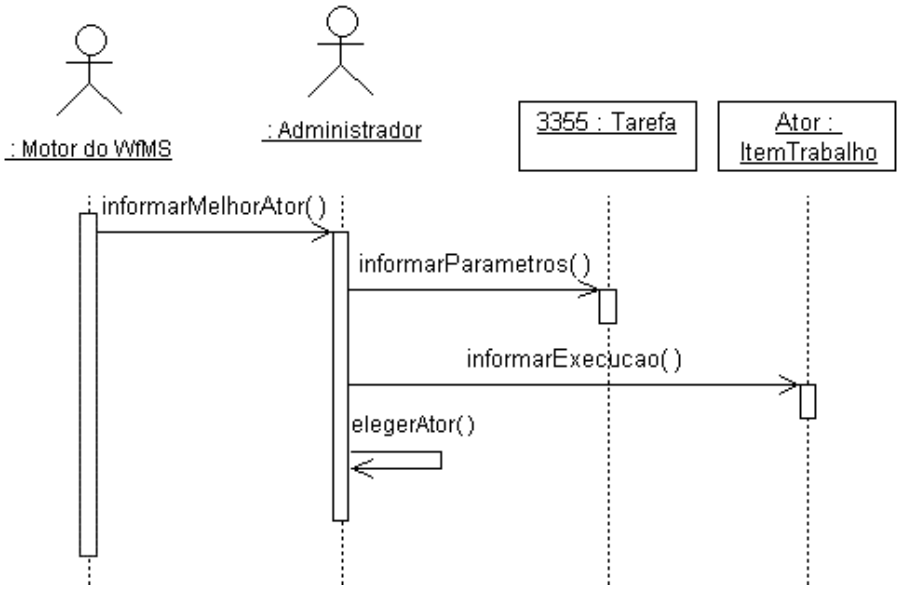

Figura B.6 – Diagrama de Seqüência Eleger Ator

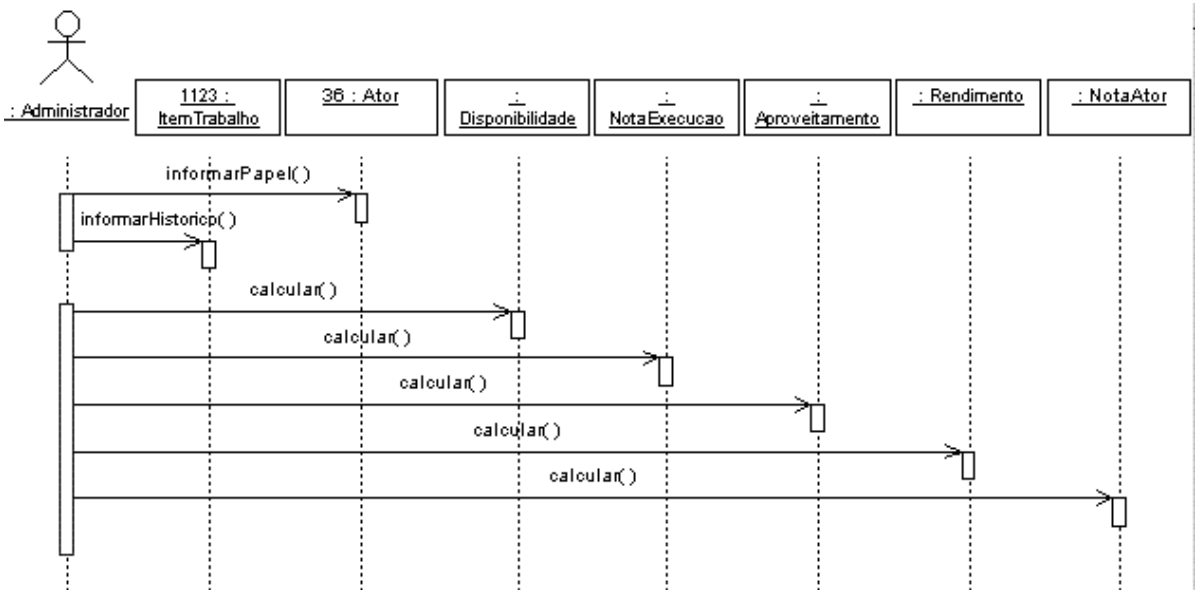

Figura B.7 – Diagrama de Seqüência - Calcular Nota

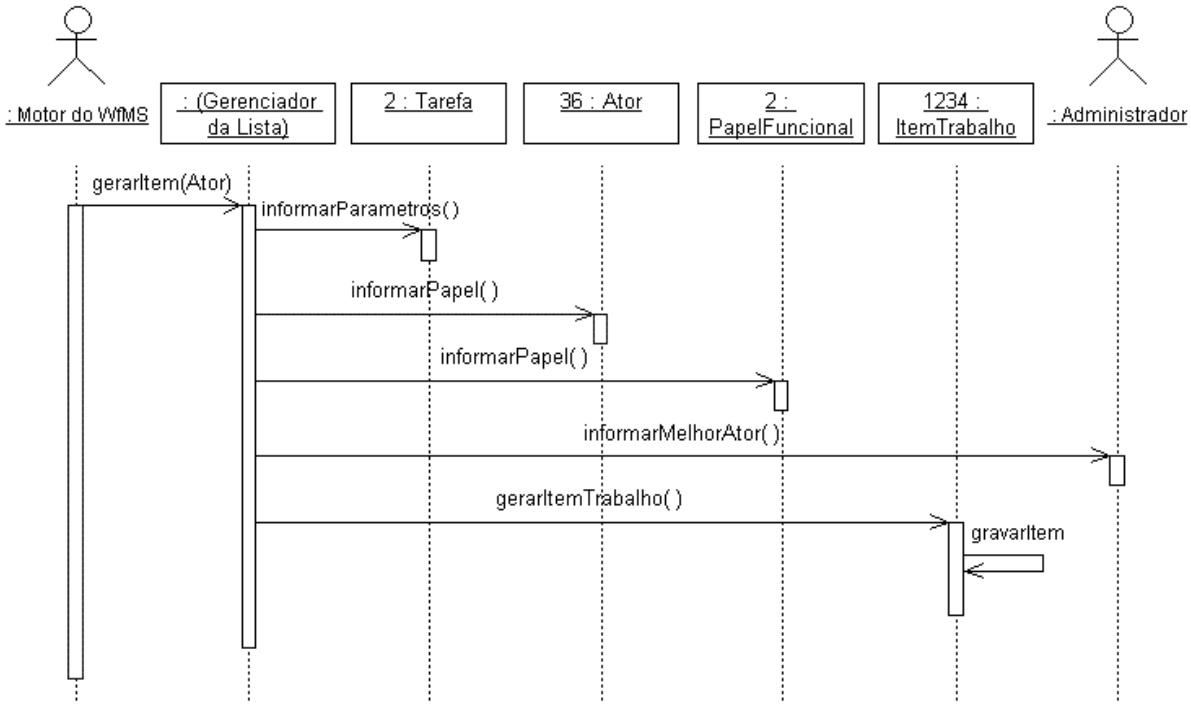

Figura B.8 – Diagrama de Seqüência – Gerar Item de Trabalho

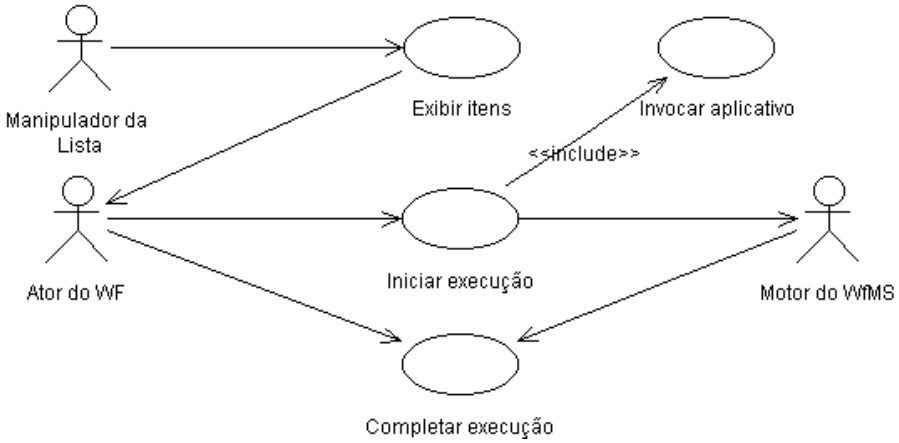

Figura B.9 – Diagrama de Caso de Uso – Executar Tarefa

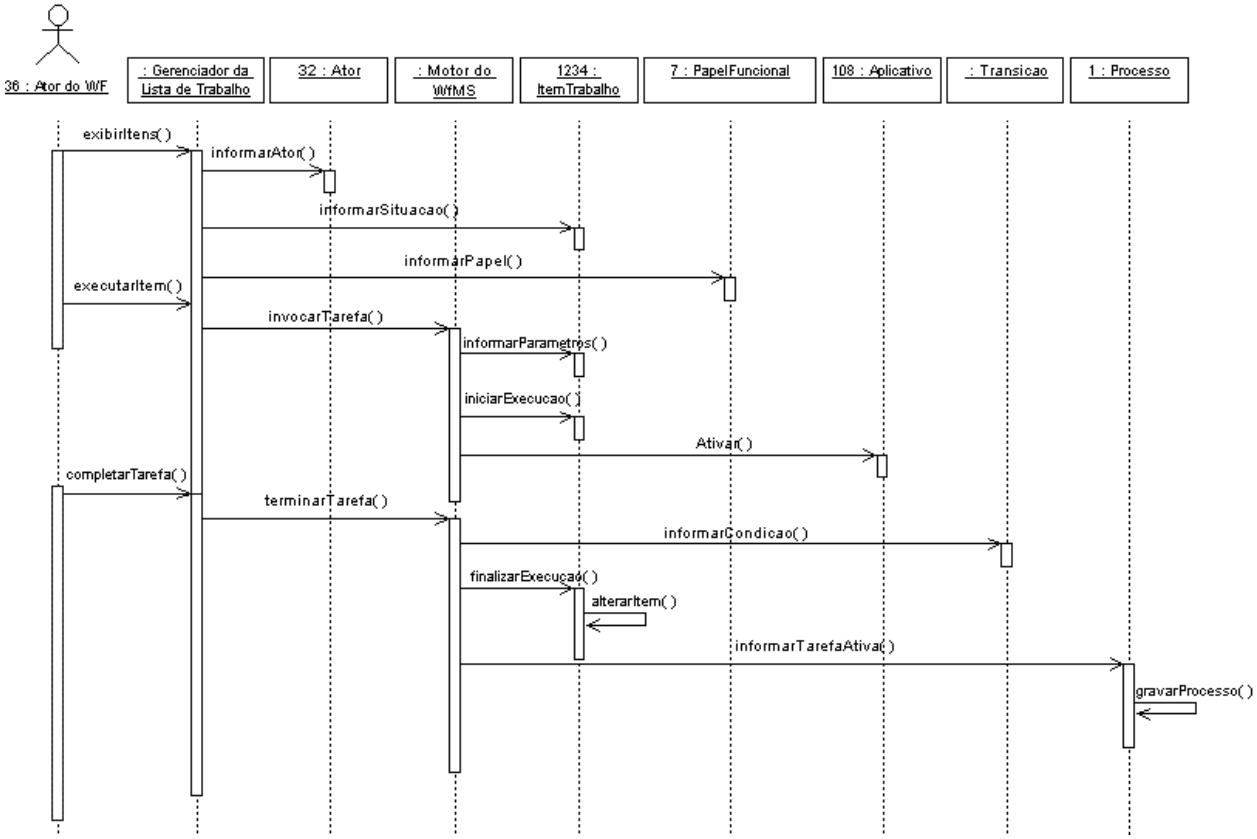

Figura B.10 – Diagrama de Seqüência – Executar Tarefa

ANEXO C – DIAGRAMA DE CLASSES DO WFFUZZY

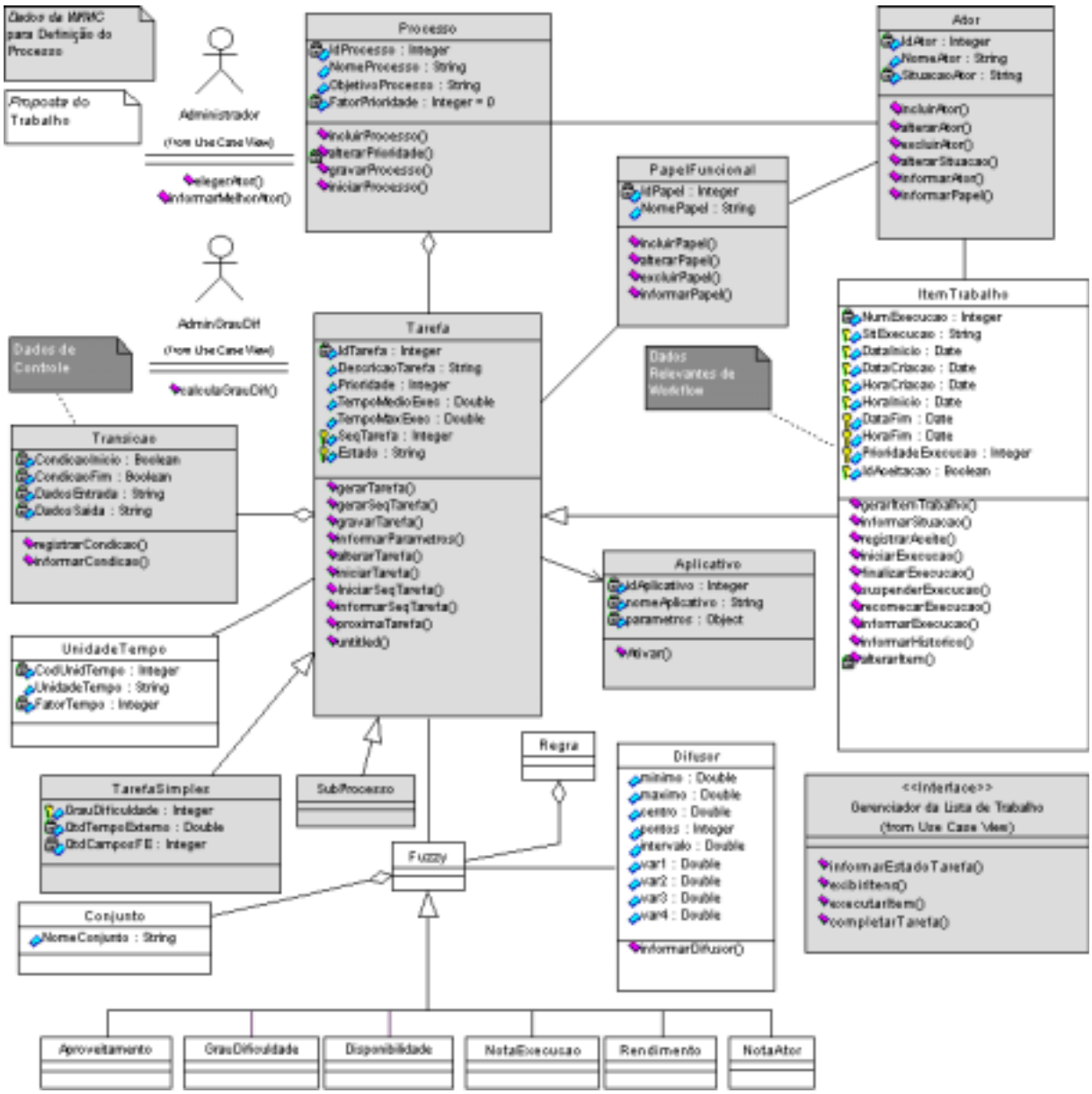

Figura C.1 – Diagrama de Classes do WfFuzzy

ANEXO D – CÓDIGO FONTE DA CLASSE FUZZY

```
public class Fuzzy
{
	//atributos da Variável 1 de entrada
	double e1_minimo,e1_maximo,e1_intervalos;
	double e1_a1,e1_a2,e1_b1,e1_b2,e1_b3,e1_c1,e1_c2,e1_c3,e1_d1,e1_d2,e1_d3,e1_e1,e1_e2;
	//atributos da Variável 2 de entrada
	double e2_minimo,e2_maximo,e2_intervalos;
	double e2_a1,e2_a2,e2_b1,e2_b2,e2_b3,e2_c1,e2_c2,e2_c3,e2_d1,e2_d2,e2_d3,e2_e1,e2_e2;
	//atributos da Variável de saída
	double s_minimo,s_maximo,s_intervalos;
	double s_a1,s_a2,s_b1,s_b2,s_b3,s_c1,s_c2,s_c3,s_d1,s_d2,s_d3,s_e1,s_e2;

	double centroAltura1;
	double centroAltura2;
	. . .                    // Pontos médios dos máximos dos conjuntos nebulosos (M-o-M)
	double centroAltura5;

	final static int NUMEROREGLAS = 25;
	final static int NUMEROVARENT = 02;
	final static int NUMEROVARSAI = 01;
	final static int NUMTOTCONJENT = 10;

	Variable variablesEntrada[];
	Variable variablesSalida[];
	Conjunto entradas[];
	Difusor difusores[];

	public int reglas[][];
	double modificadores[][];
	public double centrosAltura[][];

	public double entra[];
	public double sale[];

	public int e1, e2, s;  // Variáveis de entrada e variável de saida
```

Quadro D.1 – Código em Java da Superclasse Fuzzy

```
public Fuzzy(int code1, int code2, int cods)
{
            e1 = code1;
            e2 = code2;
            s  = cods;

            entra = new double[2];
            sale = new double[1];

            reglas = new int[NUMEROREGLAS][NUMEROVARENT+NUMEROVARSAI];
            modificadores = new double[NUMEROREGLAS][NUMEROVARENT];
            centrosAltura = new double[NUMEROREGLAS][NUMEROVARSAI];

            variablesEntrada = new Variable[2];
            variablesSalida =  new Variable[1];

            entradas = new Conjunto[10];
            difusores = new Difusor[2];

   modificadores[0][0]=1.000000;modificadores[0][1]=1.000000;
   modificadores[1][0]=1.000000;modificadores[1][1]=1.000000;
   modificadores[2][0]=1.000000;modificadores[2][1]=1.000000;
   . . .
   modificadores[24][0]=1.000000;modificadores[24][1]=1.000000;
}
// Configura a variável
public void configurar(int processo, int tarefa)
{
double universoDiscursoE1, universoDiscursoE2, intervaloDifusorE1, intervaloDifusorE2,
 difMinimo, difMaximo, centroUnivE1, centroUnivE2;
// Recupera pontos dos conjuntos da variável para a tarefa
// Variável 1 de entrada
String sqlE1 = "SELECT ConjuntoEsp.*, minimo, maximo, intervalos " + "FROM VariavelTarefa
  INNER JOIN ConjuntoEsp ON (VariavelTarefa.codVariavel = ConjuntoEsp.codVariavel) AND
  (VariavelTarefa.idTarefa = ConjuntoEsp.idTarefa) AND (VariavelTarefa.idProcesso =
  ConjuntoEsp.idProcesso) " + "WHERE (((ConjuntoEsp.idProcesso)=" + processo + ") AND
  ((ConjuntoEsp.idTarefa)=" + tarefa + ") AND ((ConjuntoEsp.codVariavel)=" + e1 + "))";
// Variável 2 de entrada
String sqlE2 = "SELECT ConjuntoEsp.*, minimo, maximo, intervalos " + . . .
// Variável de saída
String sqlS  = "SELECT ConjuntoEsp.*, minimo, maximo, intervalos " + . . .
   try
   {
      // Prepara acesso à base de dados
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      String fonteURL = new String("jdbc:odbc:workflow"); . . .
    {
         recSet = comando.executeQuery(sqlE1);
         // Recupera Universo de Discurso da Variável 1 de entrada
         recSet.next();
         e1_minimo        = recSet.getDouble("minimo");
         e1_maximo        = recSet.getDouble("maximo");
         e1_intervalos    = recSet.getDouble("intervalos");
```

Quadro D.1 – (Cont.) Código em Java da Superclasse Fuzzy

```
            // Recupera Configuração dos Conjuntos Nebulosos da Variável
            recSet.next();
            e1_a1           = recSet.getDouble("Ponto1");  // Conjunto Nebuloso1
            e1_a2           = recSet.getDouble("Ponto2");
            recSet.next();
            e1_b1           = recSet.getDouble("Ponto1");   // Conjunto Nebuloso 2
            e1_b2           = recSet.getDouble("Ponto2");
            e1_b3           = recSet.getDouble("Ponto3");
            . . .
            recSet.next();
            e1_e1           = recSet.getDouble("Ponto1");   // Último Conjunto Nebuloso
            e1_e2           = recSet.getDouble("Ponto2");
            // Cria a Variável
            variablesEntrada[0] = new Variable("Entrada1",e1_minimo,e1_maximo,e1_intervalos,
                                    (e1_maximo-e1_minimo)/e1_intervalos,5);
            entradas[0] = new Conjunto(e1_minimo,e1_a2);
            entradas[1] = new Conjunto(e1_b1,e1_b3);
            entradas[2] = new Conjunto(e1_c1,e1_c3);
            entradas[3] = new Conjunto(e1_d1,e1_d3);
            entradas[4] = new Conjunto(e1_e1,e1_maximo);
              }
        // Recupera Universo de Discurso da Variável 2 de entrada
        recSet = comando.executeQuery(sqlE2);
                    . . .
. . .       // Recupera Universo de Discurso da Variável 2 de entrada
        recSet = comando.executeQuery(sqlS);
. . .
 //Cálculo dos Difusores para as duas variáveis de entrada - tipo Singleton, com 1 ponto
 universoDiscursoE1 = e1_maximo - e1_minimo;
 if (universoDiscursoE1 <0)             {
            universoDiscursoE1 = universoDiscursoE1 * -1;
             }
 universoDiscursoE2 = e2_maximo - e2_minimo;
 if (universoDiscursoE2 <0)             {
            universoDiscursoE2 = universoDiscursoE2 * -1;
             }
  // Definição do difusor
 intervaloDifusorE1 = universoDiscursoE1 * 0.000025;
 . . .
 difMinimo = centroUnivE1 - intervaloDifusorE1;
 difMaximo = centroUnivE1 + intervaloDifusorE1;
 difusores[0] =
new Difusor(difMinimo,difMaximo,centroUnivE1,1,intervaloDifusorE1,difMinimo,difMaximo,0,0);
. . .
 difusores[1] = new   . . .
 //Cálculo do Centro-de-Máximos (C-o-M)
 centroAltura1 = s_minimo + ((s_a1 - s_minimo)/2);
 centroAltura2 = (s_b2);
 centroAltura3 = (s_c2);
 centroAltura4 = (s_d2);
 centroAltura5 = s_e2 + ((s_maximo - s_e2)/2);          }
 //Cálculo da Inferência
public void calcular()       {
            for(int i=0;i<NUMEROVARSAI;i++)         {
```

Quadro D.1 – (Cont.) Código em Java da Superclasse Fuzzy

ANEXO E – CÓDIGO FONTE DAS CLASSES GrauDif E Frm_VarTarefa

```
// Grau de Dificuldade - Atualiza tabela Tarefa
        // Configura as variáveis
        GrauDif dif = new GrauDif();
        // Executa o método de configuração para a inferência
        dif.configurar(Integer.parseInt(Proc),Integer.parseInt(Tarefa)); // proc , tarefa
        boolean grauDifOK = true;
         try {
            // Recupera variáveis de entrada para o cálculo do GrauDif
            rs = stmt.executeQuery("SELECT qtCamposFormEl,qtUnidTempoTarefa FROM Tarefa
                 WHERE idTarefa="+ Tarefa +" AND idProcesso="+ Proc +" ");
            rs.next();
            // Carrega as variáveis de entrada e calcula o GrauDif
            dif.entra[0] = Double.parseDouble(rs.getString(1));
            dif.entra[1] = Double.parseDouble(rs.getString(2));
            dif.calcular();
            msg = "Grau de dificuldade gerado!\nValor: "+dif.sale[0]+"";
            // Realiza a alteração da Tarefa com o valor calculado
            stmt.executeUpdate("UPDATE Tarefa SET grauDificuldade="'+dif.sale[0]+"' WHERE
                                idTarefa="+ Tarefa +" AND idProcesso="+ Proc +" ");

        } catch (Exception excInt) { . . .
```

Quadro E.1 – Classe Frm_VarTarefa, que define Conjuntos das Variáveis de cada Tarefa, destacando a geração do Grau de Dificuldade da Tarefa

```
public class GrauDif extends Fuzzy
{
        public GrauDif()
        {
        super(1,2,3);  // Identificadores das variáveis de entrada e da variável de saída
           // Regras de inferência sobre os Conjuntos Nebulosos
           reglas[0][0]=0;reglas[0][1]=0;reglas[0][2]=0;
           reglas[1][0]=0;reglas[1][1]=1;reglas[1][2]=1;
        . . .
           reglas[24][0]=4;reglas[24][1]=4;reglas[24][2]=4;

           // Definição dos pontos médios do método C-o-M
           centrosAltura[0][0]=centroAltura1;
           centrosAltura[1][0]=centroAltura2;
        . . .
           centrosAltura[24][0]=centroAltura5;
         }
 }
```

Quadro E.2 – Classe GrauDif e o Método que realiza a Configuração da Variável para a Inferência