UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Modelo de *WorkFlow* com Percepção de Eventos

por

LUÍS ANTÔNIO MOTA

Dissertação submetida à avaliação, como requisito
Parcial para a obtenção do grau de Mestre em
Ciência da Computação

Prof. José Valdeni de Lima
Orientador

Porto Alegre, janeiro de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Mota, Luís Antônio

Modelo de *WorkFlow* com Percepção de Eventos / por Luís Antônio Mota. - Porto Alegre: PPGC da UFRGS, 2003.

100p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Lima, José Valdeni de.

1. *Workflow*. 2. A*wareness*. 3. Trabalho cooperativo. 4. Projeto CEMT. 5. Modelo de referência da WfMC I. Lima, José Valdeni de. II. Título.

Agradecimentos

"*Não são as perdas e nem as caídas o que podem fazer fracassar nossas vidas, senão a falta de coragem para levantar e seguir a diante*" – Samael Aum Weor

O mestrado foi uma experiência nova e, pela primeira vez, quase desisti de algo antes de concluir, coisa que antes nunca aconteceria. Consegui graças ao apoio de meus amigos e de minha família e, assim sendo, os nomes que gostaria de listar aqui muitos: amigos, colegas de mestrado, da UFRGS, da UNESC, da Esmalglass e toda minha família. Sendo assim, mencionarei agora aqueles que participaram mais diretamente do êxito deste meu trabalho.

Agradeço ao prof. Valdeni, meu orientador, pela paciência e apoio quando pensei em desistir, à minha esposa Rosangela e minha filha Suelen pela compreensão e sacrifício de seus finais de semana comprometidos pelo meu trabalho; ao meu amigo Jorge Ricardo pelo seu apoio na flexibilização dos meus horários de trabalho, pelos conselhos e orientações quanto à minha organização.

Agradeço à Esmalglass, empresa onde trabalho, que me apoiou todo o tempo em tudo o que estivesse ao seu alcance, apoio fundamental para a realização de minhas pesquisas.

Agradeço aos meus colegas Thiago Telecken e Montgomery pela paciência ao me interar de suas realizações e me apoiarem no desenvolvimento do meu trabalho.

Agradeço aos meus colegas da UNESC por sua amizade e empenho por minha realização.

Muito obrigado a todos, desejo-lhes tudo de bom e muitas felicidades.

Sumário

Lista de Abreviaturas

API	Application Programming Interface
CEMT	Conception of Cooperative Environment for Editing Multimedia Documents with Workflow Technology
CIMOSA	Computer-Integrated Manufactuting – Open-Systems Architecture
CML	Chemical Markup Language
CSCW	Computer Supported Cooperative Work
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
EAD	Ensino à Distância
EDI	Eletronic Document Interchange
F-ORM	Functionality in Objects with Role Model
HTML	Hipertext Markup Language
HTTP	Hypertext Transfer Protocol
ICN	Information Control Net
IETF	Internet Engeneering Task Force
INRIA	Institut National De Recherche En Informatique Et En Automatique
ORM	Objects with Role Model
RDF	Resource Description Framework
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SIE	Sistemas de Informação de Escritórios
SMIL	Synchronized Mutimedia Integration Language
SOAP	Simple Object Access Protocol
SQL	Structure Query Language
SVG	Scalable Vector Graphics
TF-ORM	Temporal Functionality in Objects with Role Model
W3C	World Wide Web Consortium
WAPI	Workflow API and Interchange Formats
WWW	World Wide Web
WebDAV	Distributed Authoring and Versioning on the Web
WF	Workflow
WFA	Workflow Administrator
WFDL	Workflow Description Language
WfMC	Workflow Management Coalition
WFML	Workflow Modification Language
WFMS	Workflow Management System
WWW	World Wide Web
Xlink	XML Linking Language
XLL	eXtensible Linking Language
XML	eXtensible Markup Language
Xpointer	XML Pointer Language
XSL	Extensible Style Language

Lista de Figuras

Lista de Tabelas

Resumo

O objetivo deste trabalho é desenvolver um modelo de *workflow* com capacidade de percepção de eventos durante sua execução e interagir com os participantes a partir desta percepção. Um *workflow* é definido como sendo um conjunto de atividades, que podem, ou não, serem executadas simultaneamente, com alguma especificação de controle e fluxo de dados entre estas atividades, podendo representar vários processos e seus relacionamentos. Porém na área de modelagem de *workflow* existem inúmeros modelos desenvolvidos para atender uma necessidade especifica e por isso não existe um modelo definitivo, que satisfaça todas as necessidades de todos os ambientes. O modelo precisa disponibilizar expressões relacionadas aos processos, restrições temporais, trocas dinâmicas e tratamento de exceções, habilitar execução dinâmica, modelagem baseada em processos, providenciar coordenação e assistência aos usuários a cada passo da modelagem e processamento distribuído. Existe ainda a necessidade dos sistemas de *workflow* manipularem e monitorarem a informação relativa ao fluxo de trabalho para gerência-lo, coordena-lo e controla-lo mais eficientemente e para proporcionar recuperação histórica de seu fluxo permitindo assim monitoramento histórico. O principal problema das atividades baseadas em processos é que não se tem o controle total das atividades que devem ser executadas. Além disso, não se tem uma visão de quais informações estão sendo manipuladas por essas atividades em um dado instante e quem as manipula. As técnicas de modelagem de *workflow* devem ter como objetivo básico, minimizar estes problemas. Em 1993 foi criado o *WfMC* (*Workflow Management Coalition)* com o objetivo de padronizar um modelo de *workflow* e com isto alcançar uma interoperabilidade entre os vários usuários da tecnologia de *workflow*, e por isto este trabalho baseia-se no modelo de *workflow* da *WfMC*. Utilizando um recurso previsto no modelo da *WfMC*, os atributos extendidos, e utilizando recursos implementados diretamente na máquina de *workflow*, como monitoramento das atividades, ferramentas de comunicação entre os participantes do *workflow*, regras de compartilhamento de resultados intermediários, possibilidade de troca de informações durante a execução das tarefas por vários participantes ao mesmo tempo, foi obtida uma proposta de utilização do modelo da *WfMC*, sem nenhuma alteração, apenas utilizando-se dos recursos mencionados acima, obtendo o ambiente de *workflow* com capacidades de percepção de eventos conforme o objetivo deste trabalho e ainda com capacidade de cooperação na execução das tarefas como conseqüência da capacidade de percepção, compartilhamento de resultados intermediários e possibilidade de comunicação entre os participantes. Sendo esta proposta implementada em um software para validação da mesma.

Palavras-chaves: *workflow*, *awareness*, trabalho cooperativo, projeto CEMT, modelo de referência da *WfMC*

TITLE: “MODEL OF WORKFLOW WITH PERCEPTION OF EVENTS”

Abstract

The aim of this work is to develop a workflow model with capacity of perception of events during execution and to interact with the participants starting from this perception. A workflow is defined as being a group of activities that can, or no, be executed simultaneously, with some control specification and flow of data among these activities, could represent several processes and their relationships. However in the area of workflow modelling countless models exist, developed to assist a need specifies and for that a definitive model doesn't exist, that it satisfies all of the needs of all the atmospheres. The necessary model to make available expressions related to the processes, temporary restrictions, dynamic changes and treatment of exceptions, to enable dynamic execution, modelling based on processes, to provide coordination and attendance to the users to each step of the modelling and distributed processing. Still exists the need of the workflow systems manipulate and they monitor the relative information to the work flow to manage, to coordinate and to control the work more efficiently and to provide historical recovery of the work flow allowing like this to monitor historical. The main problem of the activities based on processes is that the total control of the activities is not had that should be executed. Besides, they are not had a vision of which information are being manipulated by those activities in a die instant and who manipulates them. The techniques of workflow modelling should have as basic objective, to minimize these problems. In 1993 WfMC was created (Workflow Management Coalition) with the objective of standardizing a workflow model and with this to reach an interoperability among the several users of the workflow technology, and for this work bases on the model of workflow of WfMC. Using a resource foreseen in the model of WfMC, the extended attributes, and using resources implemented directly in the it conspires of workflow, as monitoring of the activities, communication tools among the participants of the workflow, rules of sharing of intermediate results, possibility of change of information during the execution of the tasks for several participants at the same time, it was obtained a proposal of use of the model of WfMC, without any alteration, just being used of the resources mentioned above, obtaining the workflow environment with capacities of in accordance perception of events the objective of this work and still with cooperation capacity in the execution of the tasks as a consequence of the perception capacity, sharing of intermediate results and communication possibility among the participants. This proposal was implemented in software for validation of the same.

Keywords: workflow, awareness, cooperative work, project CEMT, model of reference of WfMC

1 Introdução

A busca por novas metodologias de ensino tem resultado no crescimento da utilização de programas de computador na educação e, conseqüentemente, promovido as pesquisas em desenvolvimento de programas educacionais.

A necessidade de investimento nesta área no Brasil torna-se evidente, considerando-se a extensão do território nacional e a crescente demanda por ensino a distância. Com base nestas considerações, os sistemas de ensino/aprendizagem devem ser objetos de estudo e aprimoramento constante, tornado-se uma ferramenta de acesso irrestrito e eficiente para o desenvolvimento dos indivíduos que dela se utilizem, como ferramentas bem fundamentadas e de baixo custo.

Este trabalho se enquadra com os objetivos do projeto CEMT "*Conception of Cooperative Environment for Editing Multimedia Documents with Workflow Technology"* [OPE 2001] [TEL 2001] no desenvolvimento de um ambiente para edição e execução de documentos multimídia. Um destes ambientes desenvolvido pelo projeto é o MADEUS [INR 2002] [INR 2002a], um ambiente de edição e apresentação de documentos multimídia.

Um modelo de percepção de *workflow* de execução pode ser entendido como um conjunto de regras, ferramentas e agentes que monitoram, explicam e demonstram como um documento deve ser utilizado, proporcionando a quem o utiliza um "Mentor à Distância" e ao gestor do processo um monitoramento do desempenho, da utilização e possibilidade de interação entre todos os participantes. Este conjunto de regras, ferramentas e agentes, para suporte à percepção [KIR 2001], proporcionam uma interação antes obtida in loco, agora de forma assíncrona no tempo e no espaço, interação, esta, de grande valor para o êxito de cursos à distância. Desta forma o modelo seria uma composição de linhas gerais, que abrangem determinadas necessidades, as quais podem ser ou não implementadas de acordo com a finalidade do documento multimídia utilizado, dando ao autor a disponibilidade de recursos para ministrar seu conhecimento a todo aquele que utilize seu curso e dando ao "aluno" a oportunidade de ter um "amparo" e um mecanismo de auto-avaliação.

A utilização do computador no ensino tem crescido em larga escala, o que pode ser observado pelo surgimento dos laboratórios de multimídia nas escolas e o aumento do número de programas educativos desenvolvidos para esse fim. A origem deste trabalho baseia-se na expectativa de que se obtenha com a utilização da Internet e do uso do resultado do mesmo aplicado à ferramentas como o MADEUS, BIZANCE [TEL 2001] ou THOT [THO 2001] uma maior facilidade de disponibilização pública do material educativo já desenvolvido e em desenvolvimento e de sua integração com outros elementos objetivando a educação a distância, colaborando com o desenvolvimento de ferramentas multimídias de edição e execução locais ou remotas, que auxiliem na formação dos indivíduos através de ensino a distância.

Os sistemas de *workflow* são o objeto de estudo deste trabalho. Estudou-se sobre CSCW, *workflow* e percepção de eventos e sobre as questões que envolvem estas áreas bem como algumas ferramentas que tratam estas questões, para também poder sugerir uma definição de *workflow* adequada às necessidades de percepção.

O objetivo principal é a obtenção de um modelo de *workflow* com capacidade de expressar e executar conceitos de percepção de eventos, monitorando e interagindo com o autor e com os agentes do processo modelado, compatível com o modelo de *workflow*

da WFMC "*Workflow Management Coalition*" o que garante a integração com a tecnologia de *workflow* usada no projeto CEMT.

Este trabalho está assim organizado: Capítulo 2 trazendo uma revisão geral das tecnologias envolvidas na área de trabalho cooperativo, *workflow*, projeto CEMT e WfMC; em seguida o Capítulo 3 trata especificamente o tema awareness, a capacidade de percepção que este trabalho deve alcançar; no Capítulo 4 apresenta-se o estado da arte sobre os modelos de *workflow*; no Capítulo 5 tem-se a proposta deste trabalho, sendo que o desenvolvimento do trabalho é apresentado no Capítulo 6. Por fim no Capítulo 7 (conclusão) são apresentadas as conclusões deste trabalho e as propostas de trabalhos futuros.

2 Trabalho Cooperativo

De acordo com [SOU 96], o termo “trabalho cooperativo” originou-se ainda no século XIX, quando foi utilizado por economistas para designar o trabalho envolvendo vários autores. O termo CSCW *“Computer Suported Cooperative Work”* demonstra claramente os objetivos a serem alcançados por esta tecnologia, a edição de documentos de forma colaborativa e ou realização de outras tarefas com um fim comum entre os participantes de um grupo, é uma atividade que envolve a participação e contribuição dos vários integrantes os quais precisam ter ferramentas que os auxiliem a organizar, comunicar, editar e executar as tarefas, de forma colaborativa, necessárias ao objetivo proposto. Na execução de um trabalho em grupo de forma interativa, os membros cooperam entre si para chegar a um objetivo comum, cada um contribuindo com a sua parte para o produto final. Para isso o ambiente de trabalho, suportado por computador precisa, prover as ferramentas necessárias para a realização da comunicação entre os componentes do grupo e do compartilhamento das informações, propiciando assim a execução de forma cooperativa.

No contexto do CSCW enquadram-se as aplicações de tecnologia de informação que suportam grupos de trabalho cooperativos, abordando a concepção e desenvolvimento de um suporte informatizado ao trabalho em grupo. O objetivo desta área de pesquisa é, por um lado, estudar os aspectos cognitivos e sociais do processo de cooperação e, por outro lado, desenvolver um suporte eficaz para o trabalho em grupo.

A idéia por detrás do trabalho cooperativo é fazer com que a produção do grupo seja maior do que a produção de cada um dos membros individualmente. Este ideal representa um aumento significativo de produtividade, o que vem sendo apontado como um importante fator para a sobrevivência das empresas no mercado. A palavra chave aqui é a competitividade. A empresa (ou grupo) mais produtiva mantém-se competitiva por mais tempo no mercado e assim tem mais chances de sobrevivência e de alcançar uma posição de destaque. A seguinte citação de [SOU 96] resume bem esta situação: “As soluções baseadas em *groupwares* estão emergindo para fornecer maior competitividade às organizações”. Por esta citação fica claro o papel dos *groupwares*: agilizar o trabalho em grupo, de forma a atingir o tão esperado aumento na produtividade.

Dentro dessa realidade, vários fatores têm sido determinantes para a área de CSCW. A já referida necessidade por resultados rápidos tem sido a grande incentivadora deste processo de formação de grupos. Além desse, outros fatores têm também impulsionado a pesquisa em CSCW nos últimos anos, onde se destaca a adoção e disseminação de redes de computadores nos mais variados ambientes de trabalho; a necessidade de compartilhamento de recursos e informações; e a adoção de sistemas distribuídos, que podem cooperar para processar uma transação de negócios e estão tomando o lugar de sistemas tradicionais de controle centralizado.

Entretanto, para que um *groupware* seja aceito, ele não pode mudar radicalmente a forma como as pessoas trabalham, sob o risco de não ser usado a contento, causando frustração, no lugar de ganhos em qualidade e produtividade. Tampouco pode ser projetado para ser imposto sobre um grupo, sem levar em consideração questões relacionadas à organização hierárquica das empresas e ao papel representado pelos participantes do grupo [DIE 96]. Estas questões são determinantes para que um *groupware* atenda seus objetivos. Observando o cotidiano de empresas é comum ver

membros com funções diferenciadas dentro do grupo, como “gerente” e “chefe de seção”, denotando uma organização hierárquica dentro do mesmo. Esta organização pode seguir diversos modelos, com vários níveis. Pode ser totalmente plana, com todos os membros gozando dos mesmos direitos; pode ser plana, mas com um moderador responsável por unir todas as contribuições dos demais membros; ou ainda pode ser realmente hierárquica, com uma ou mais camadas (gerente, chefe, subchefe). Cada grupo pode adotar um modelo diferente de organização, adaptado às suas necessidades. A organização utilizada pelo grupo é peça importante na sua dinâmica, uma vez que seus membros estão habituados a trabalhar neste sistema, e um *groupware* deve saber respeitar e se adaptar a organização adotada pelo grupo.

A organização hierárquica do grupo faz parte dos aspectos sociais estudados na área de CSCW. Junto com a organização são estudados também os modelos cognitivos que podem ser adotados pelo grupo. Um modelo cognitivo consiste nas fases pelas quais o processo de trabalho em grupo se divide, e engloba a descrição do comportamento dos autores no decorrer do trabalho. Da mesma forma que a organização, existem vários modelos cognitivos que podem ser adotados pelo grupo e vão representar a maneira como o grupo trabalha para atingir seu objetivo. Grande parte desses modelos apresenta, de alguma forma, fases para o planejamento, a coordenação e a negociação das atividades. No planejamento são definidas as atividades que devem ser executadas para que o objetivo do grupo seja atingido, e o papel de cada membro no decorrer do trabalho. A coordenação procura evitar que tarefas redundantes sejam executadas e permite que cada membro possa gerênciar suas atividades de acordo com os demais. E durante a negociação, busca-se resolver os conflitos gerados na realização das atividades, muitas vezes, através simplesmente da comunicação entre os membros [DIE 96].

O suporte a estes aspectos sociais dentro de um *groupware* é vital, pois a essência deste tipo de sistema é justamente o elemento humano, e onde há elementos humanos trabalhando em conjunto, está também toda a complexidade de relacionamentos entre as pessoas, comuns a qualquer grupo, informatizado ou não. Daí a importância do estudo destes aspectos dentro de CSCW, buscando conceitos da psicologia e sociologia. Um *groupware* deve fazer uso destes conceitos e oferecer suporte a aspectos sociais. Ele deve agir como a porta de acesso de um usuário aos seus colegas. Esta, inclusive, é apontada como a diferença entre *groupwares* e *softwares* comuns: ao contrário dos outros *softwares*, que procuram esconder um usuário dos demais, os *groupwares* devem acentuar o ambiente multi-usuário, coordenando e orquestrando as atividades, de modo que os usuários possam ver um ao outro e resolver os possíveis conflitos, como um grupo coeso.

Dessa forma, vários requisitos para que um *groupware* seja bem aceito e atenda a seus usuários podem ser apontados [DIE 96], tais como: ser confiável, dar um correto suporte à informação compartilhada, facilitar a cooperação, fornecer uma percepção da interação do grupo, não impor práticas que causem mudanças radicais na forma de trabalho do usuário, ser composto preferencialmente por aplicações menores e inter-relacionadas, e suportar ambientes heterogêneos e abertos. Os aspectos sociais também devem ser contemplados, pois várias são as condições que podem fazer um grupo trabalhar melhor, como, por exemplo, uma boa identificação da tarefa como significativa e com resultados visíveis; a existência de *feedback* sobre o trabalho; a comunicação ente os usuários, tanto para a troca de opiniões, sugestões e comentários, quanto para a coordenação das atividades e para a negociação.

A comunicação entre os usuários merece especial destaque. A troca de opiniões, comentários e anotações são importantes para o crescimento do trabalho em grupo.

Quando este grupo encontra-se geograficamente próximo, esta comunicação pode ser informal, através de telefonemas e até dos bate-papos no intervalo para o café, não exigindo do *groupware* mecanismos sofisticados. Entretanto, quando os participantes estão geograficamente dispersos, esta comunicação informal torna-se difícil (ou impossível), à medida que se torna tarefa do *groupware* fornecer suporte a esta comunicação.

A adequação de todos esses pontos expostos acima vai determinar o sucesso de uma ferramenta desenvolvida para suportar o trabalho cooperativo, um sistema de *workflow* deve ter suporte em seu modelo e ou em sua máquina de execução a estes fatores de usabilidade e comunicação, para proporcionar ao usuário um ambiente cognitivo que lhe permita executar e auxiliar na elaboração execução de tarefas.

2.1 As Tecnologias

Além do reconhecimento e aplicação de áreas da ciência do comportamento, como Psicologia e Sociologia, o desenvolvimento de um suporte eficaz ao trabalho cooperativo, devido a sua complexidade, envolve também várias tecnologias da própria Ciência da Computação, tais como Redes de Computadores, Banco de Dados, Sistemas Operacionais, Interação Homem-Máquina, Multimídia e Hipermídia e Agentes Inteligentes [SOU 96] e [DIE 96] e um sistema de *Workflow* adequado às necessidades do trabalho em grupo, com capacidades de percepção de eventos e colaboração na execução das tarefas. Sendo este o objetivo deste trabalho, vamos listar algumas características destas tecnologias.

2.1.1 Agentes Inteligentes

O efetivo uso de Agentes Inteligentes em sistemas tem aparecido principalmente nesta última década. Em CSCW, o seu uso concentra-se principalmente na utilização de agentes para a realização de tarefas, que seriam tediosas ao elemento humano, mas que são importantes para a atividade cooperativa, como a criação de boletins informativos e a notificação da presença de participantes [DIE 96].

2.1.2 Hipertextos e Multimídia

As tecnologias de Multimídia e Hipertextos são de grande interesse para a área de CSCW. O uso da multimídia fornece um meio de interação mais natural entre os participantes, através de recursos como músicas, imagens e vídeo, que são elementos de comunicação universal. O problema no uso destes objetos multimídia encontra-se principalmente na digitalização destas mídias. Este processo pode ocupar grande espaço em disco e memória, mesmo com o uso de algoritmos de compressão, o que dificulta bastante o seu transporte por redes de comunicação de baixo desempenho. Já o enfoque de hipertextos em CSCW permite a criação e interligação de fragmentos de informação [DIE 96].

O uso destas tecnologias está intimamente ligado à tecnologia de Interface Homem-Máquina, à medida que estas são utilizadas para a criação de interfaces mais convenientes e funcionais para *groupwares*, especialmente quando um grande número de informações a ser compartilhadas está disponível.

Hipertexto ou hiperdocumento pode ser conceituado como sendo um texto ou documento não-linear, formado por um conjunto de nós e uma rede de ligações entre eles. Cada nó pode ser um trecho de informação do documento, que representa um conceito ou idéia, enquanto que as ligações conectando estes nós representam os relacionamentos entre eles. Para [SOU 96], hipertextos assemelham-se à redes de nós, interligados, denotando a associação de informações entre estes nós.

O termo hiperdocumento é usado, normalmente, de forma a abranger hipertextos somados a recursos multimídia, e pode ou não ser autocontido (possuir apenas ligações que referenciam nós que o compõem).

Usando esta tecnologia criou-se o conceito de hiperdocumentos ativos. Um hiperdocumento é dito ativo quando deixa de ser um objeto estanque, acessado apenas para leitura, para se tornar uma ferramenta ativa, que interage com o usuário e com o ambiente onde está sendo utilizado. Este tipo de hiperdocumento é utilizado, na área de CSCW, em sistemas de autoria cooperativa, que permitem a interação entre vários autores e reagem as atividades de cada um destes autores, pela propagação de alterações e troca de mensagens.

A partir de meados da década de 90, os hipertextos têm sido alvo de atenção mundial, devido a criação e popularização da WWW, que introduziu o hipertexto como uma das principais formas de utilização da Internet. Citando [DIE 96]: "*É cada vez maior o interesse de pesquisadores da área de CSCW em desenvolver mecanismos de colaboração utilizando o ambiente WWW, devido a sua grande difusão em todo o mundo e capacidade de trabalho em ambientes heterogêneos*".

2.1.3 Interface Homem-Máquina

Para [DIE 96], a utilização de tecnologias de Interface Homem-Máquina se dá basicamente através da interface do sistema. Esta deve preferencialmente encorajar a cooperação, dando ciência das atividades dos demais participantes e podendo se adaptar aos requisitos do grupo. Isto porque um dos objetivos que deve ser perseguido no desenvolvimento deste tipo de sistema é a praticidade da interface. A interface de um *groupware* deve ser útil, enfatizando a interação e o compartilhamento das atividades. Em um *groupware*, a interface deve ser vista principalmente como um caminho de acesso aos demais membros do grupo. Lembrando sempre que justamente a percepção dos demais é um dos pontos que diferencia um *groupware* de *softwares* comuns.

2.1.4 HTML – Hiper Text Markup Language

Este padrão, hoje amplamente difundido é um formato que descreve como uma página *Web* deve ser exibida e não oferece nenhuma descrição dos dados. O HTML é uma aplicação derivada do padrão SGML, mas oferece um conjunto limitado de *tags*. O SGML possui um grande número de *tags* e oferece a possibilidade da criação de qualquer conjunto de *tags*. O HTML descreve o formato de apresentação, não descreve a estrutura de dados e apresenta um número limitado e não extensível de *tags* é por isso inadequado para gerênciamento de grande volume de dados, assim não oferece a funcionalidade requerida pelo comércio eletrônico e outras aplicações que necessitem intercâmbios de informações.

2.1.5 CSS – Cascading Style Sheets

CSS é a especificação de folhas de estilo utilizadas para formatar a apresentação das páginas HTML, enquanto que o XSL formata a apresentação dos documentos XML [CSS 2003], entretanto podem ser utilizados ao mesmo tempo para apresentar dados XML e HTML de acordo com a aplicação.

CSS é um padrão desenvolvido pela W3C, sendo que o XSL também é desenvolvido pela W3C, apesar de terem o mesmo propósito a W3C desenvolve estes dois trabalhos devido à necessidade diferenciada dos documentos XML e HTML, assim um modelo complementa o outro. A W3C trabalha duro para garantir a

interoperabilidade entre estes dois modelos e para que os dois apresentem as mesmas características e possibilidades.

2.1.6 SGML – Standard Generalized Markup Language

O SGML é de difícil adoção na *Web* devido a sua complexidade, sendo difícil seu suporte por *browsers web*, por isso não existem estilos amplamente difundidos. Para uso na *web* é convertido para HTML perdendo muito da inteligência do documento original, impedindo a sua reutilização, intercâmbio e automação.

O SGML oferece a possibilidade da criação de qualquer conjunto de *tags*. É um padrão internacional, definido em 1986, para formato de texto e documentos. Sendo popular em organizações que precisam criar e gerênciar grandes volumes de documentos. Padrão adotado, entre outros, pela indústria aeroespacial, automotiva, de telecomunicações e de software. Compatível com padrões atuais e futuros, não proprietário, não se torna obsoleto.

O SGML possui as seguintes limitações na *web*:

- Falta de suporte pelos *browsers* mais populares
- Uso SGML na *Web* envolve a tradução para HTML, e isto causa:
  - Perda de informações
  - Intercâmbio e automação de dados ficam muito mais difíceis
  - Impossível reconstruir o arquivo original a partir do arquivo HTML
  - Falta de suporte para folhas de estilo

2.1.7 XML – eXtensible Markup Language

XML é uma linguagem de marcação, isto é contém basicamente duas coisas: dados e *metadados*. Os *metadados* são informações extras que adicionam um contexto ou um significado aos dados em si [LAM 2003]. Um exemplo simples seria, por exemplo, a frase “Meu gato se chama Eddy”, as pessoas que lêem esta frase sabem que gato é uma espécie de animal e Eddy é seu nome, mas para um computador não existe este significado explicito, assim adicionamos os *metadados* para dar significado aos dados, isto com sintaxe XML, desta forma:

```
<frase>
Meu <animal>gato</animal> se chama <nome>Eddy</nome>
</frase>
```

Para que o XML saiba como se estrutura o documento e como se relacionam as *tags* criadas e dar significado aos dados, é necessário especificar uma DTD com estas informações [MAC 2003].

O XML pode ainda utilizar várias DTD´s ao mesmo tempo, com a utilização dos *namespaces*. Desta forma o XML torna possível uma infinidade de formatos para intercâmbio de dados. O XML pode fazer pelos dados o que a linguagem Java fez pelos programas, que é tornar os dados independentes de plataforma e fornecedores de software. O XML é uma forma de SGML para a *Web* apresentando formato para descrição de dados estruturados, sendo também otimizado para entrega de informações SGML através da *Web*. Os dados estruturados utilizando XML são independentes dos aplicativos utilizados ou fornecedores de software. O XML suporta praticamente todas as funcionalidades mais difundidas do padrão SGML, sendo bem mais simples, por exemplo, a mesma especificação SGML de 500 páginas em especificação XML são apenas 33 páginas [XML 2003]. Os documentos XML são em formato texto, semelhante ao HTML em muitos aspectos, mas permite um número ilimitado de *tags*, cada uma delas indicativa, não de como algo deve ser exibido, e sim do que significa. O XML é um padrão em desenvolvimento por um grupo de trabalho do W3C (*WWW Consortium*), este grupo é constituído por cerca de 14 empresas e organizações, entre elas Adobe, Microsoft, HP, Netscape. Assim o XML é uma forma de SGML adaptado a *web*, que foi criado para tornar possível a entrega de informações SGML na *Web*, eliminando as limitações do padrão SGML ao mesmo tempo em que oferece todos os seus benefícios.O XML define o conteúdo (dados) e as *tags* descrevem os dados. As *tags* são definidas pelo criador do documento e a apresentação é definida por folhas de estilo, desta forma os dados são separados da apresentação e do processamento.

Para interpretar os documentos XML existem duas APIs que podem ser utilizadas: DOM (*Document Object Model*) e SAX (*Simple API for XML*) [WIL 2003]. DOM é uma interface de programação para documentos HTML e XML e define a maneira como o documento pode ser acessado e manipulado. DOM é um padrão definido pelo W3C que cria uma visão em árvore do documento XML e tem como objetivo fornecer uma interface de programação padrão que pode ser usada em diversos ambientes e aplicações [MAC 2003]. Duas maneiras de se visualizar documentos XML: Se o *browser* entende o padrão XML,o documento pode ser enviado diretamente para ele, ou então pode-se usar uma folha de estilo para transformar o arquivo XML em algo que o *browser* entenda.

Os dados semi-estruturados podem ser descritos por um esquema. Um esquema define características de classe de objetos e podem ser divididos em Esquemas sintáticos e Esquemas conceituais. Esquemas sintáticos são esquemas que descrevem classes de documentos que são extremamente sintáticas (por exemplo, XML). Já os Esquemas conceituais descrevem classes de documentos que possuem conceitos e relações entre estes conceitos.

Existem vários esquemas para descrever dados semi-estruturados, entre os quais podemos citar a DTD, XML *Schema* e RDF *Schema*.

A consulta aos documentos XML podem ser feitas através das linguagens de consulta como: XML-QL (Xquery) baseada no banco de dados SQL, XQL baseada na linguagem de navegação Xpath, LuceneXML [XML3], TransQuery [XML3], SODA2 [XML3], NIAGARA [XML3], XPERT [XML3], etc.

2.1.8 DTD – Document Type Definition

O padrão comum (DTD - *Document Type Definition*) cobre o conjunto de *tags* a serem adotadas e a gramática de uma linguagem de marcação (*markup language*) [WIL 2003], a DTD define:

- As *tags* de cada documento
- Quais *tags* podem conter outras *tags*
- O número e seqüência das *tags*
- Os atributos que as *tags* podem ter e seus valores
- Para criar uma DTD é necessário analisar o documento que será representado quanto à:
  - Quais elementos ocorrem no documento?
  - Como se relacionam?
  - Como será a interação dos usuários com eles?

Por isso quanto mais amplo o escopo maior a complexidade e mais difícil conseguir a concordância de todos os envolvidos. As empresas que desejarem usar XML para troca de informação com outras empresas precisam descobrir uma maneira simples de encontrar a informação de que necessitam sobre os esquemas (DTDs), documentos e processos de negócios que outras empresas suportam, este problema pode ser minimizado pela criação de portais especializados em localizar, gerênciar e informar sobre o padrão XML, XLS e os modelos de informação usados em milhares de aplicações.

Um DTD é como um formulário padrão que é preenchido. Os dados podem vir de uma consulta a um banco de dados, de uma busca em documentos ou pesquisa em um catálogo on-line. Quando o formulário estiver preenchido, ele pode então ser enviado a quem solicitou o documento.

A DTD é o padrão recomendado pela W3C para descrever documentos XML, que vem sendo amplamente utilizado na WEB, principalmente para fins de documentação. Apesar disso a DTD é ainda pouco utilizada para fins de validação. Isso se deve ao fato de que a DTD é uma estrutura herdada de SGML e apesar de ter sido simplificada, sua sintaxe é diferente da sintaxe XML, o que dificulta o trabalho do usuário.

2.1.9 XSL - eXtensible Style Language

XSL é uma recomendação da W3C, sendo compatível com CSS (*Cascading Style Sheet*), tem capacidade de reordenação da informação sem consultar o servidor *Web* e possui suporte aos formatos impressos e on-line.

Foi criada para permitir uma forma prática de exibir dados XML em um navegador, ou seja, permite apresentar os dados de uma maneira mais inteligível. Podemos usar a CSS em conjunto com a XLS para apresentar os dados de um arquivo XML. Assim XML contém os dados e o XSL e CSS a apresentação, a combinação destes gera o HTML [EXT 2003].

Podemos também dizer que XSL contém informações que permitem converter um documento XML para outro formato HTML, RTF, etc, permitindo assim múltiplas apresentações de um mesmo documento.

2.1.10 XSLT - eXtensible Style Language Transformations

É uma linguagem de transformação baseada no XML. Existem atualmente muitos processadores XSLT e muitas linguagens de programação são suportadas por estes processadores [WIL 2003]. Um processador XSLT toma dois arquivos para entrada, um é a fonte XML a ser transformada e o outro é uma folha de estilo XSLT que define a transformação.

2.1.11 XPDL – XML Process Definition Language

XPDL é um dos padrões desenvolvido pela WfMC, que junto com os outros padrões da WfMC, provê um conjunto de ferramentas para implementação de gerência de processos de negócios e máquinas de *workflow*.

A WfMC tem identificado 5 (cinco) interfaces funcionais para um serviço de *workflow* como parte do programa de padronização. A especificação XPDL é parte integrante da documentação relacionada ao "Interface 1 – *Supporting Process Definition Import and Export*" da WfMC. Esta interface inclui um *metamodelo* para a descrição da definição do processo (XPDL) e também um XML *schema* para a definição do processo de intercambio [WOR 2003].

O XPDL foi criado para possibilitar a troca de informações entre usuários de sistemas de *workflow*. A escolha da base XML foi feita devido à interoperabilidade e extensibilidade de tal plataforma. Foi definido um modelo de *metadados*, contendo as entidades comumente utilizadas em processos de *workflow* de forma que todos os sistemas que seguirem as premissas do modelo definido pela WfMC serão compatíveis entre si.

A chave do XPDL é a sua extensibilidade que o capacita a manusear informação utilizada por uma grande variedade de diferentes ferramentas.

O conceito de "Definição de Processo" é definido pela [WOR 99] como sendo a representação de um processo de negócio na forma que apóie manipulação automática, como modelagem ou representatividade por um sistema de gerência de *workflow*. A definição do processo consiste em uma rede de atividades e suas relações, critérios que indiquem início e fim dos processos e informações sobre as atividades individuais, como seus participantes, aplicações de TI associadas e dados.

2.2 Groupwares

Os *groupwares* podem ser classificados principalmente usando dois critérios: o temporal e o espacial [DIE 96] e [SOU 96]. Estes dois aspectos sugerem uma representação através de uma matriz espaço/tempo, que divide o trabalho cooperativo em quatro categorias.

TABELA 2.1 Matriz de Classificação Espaço/Tempo

		TEMPO	
E		***Síncrono***	***Assíncrono***
S ***P*** ***A*** ***Ç*** ***O***	***Mesmo local***	Interação Síncrona (face-à-face)	Interação Assíncrona
	Local diferente	Interação Síncrona Distribuída	Interação Assíncrona Distribuída

Na tabela 2.1 acima, a dimensão do espaço trata da localização física dos participantes: mesmo local ou locais diferentes (remoto ou distantes). Já a classificação temporal trata do momento em que os participantes trabalham, e se divide em síncrona e assíncrona. Em uma interação síncrona a presença dos usuários cooperantes é requerida, o que não ocorre na interação assíncrona, onde os usuários trabalham em diferentes momentos. O que define uma interação assíncrona é a presença de uma defasagem entre uma ação e sua percepção pelos demais autores. Tipicamente em ambientes síncronos, os usuários interagem simultaneamente sob um mesmo conjunto de dados, através de um espaço de informações compartilhadas. Nestes ambientes o maior problema fica por conta do controle de concorrência entre o acesso dos vários co-autores. Outro problema é como será feita a difusão das ações de um autor para os demais. Questões como custo e eficiência devem ser analisadas. Além disso, também são possíveis ambientes híbridos, que suportem ambos os modos de cooperação temporal.

A tabela 2.2 a seguir distribui exemplos de *groupwares* nas 4 dimensões definidas pela matriz espaço/tempo da tabela 2.1, de acordo com suas características, onde pode ser observado a presença da edição cooperativa não só em uma categoria, mas em todas as quatro. Os sistemas de edição cooperativa cobrem todas as categorias, porque existem em diversas situações, baseadas nas necessidades dos autores e do campo de aplicação dos documentos gerados.

TABELA 2.2 Exemplos de Groupware Quanto à Classificação

		TEMPO	
		Síncrono	***Assíncrono***
ESPAÇO	***Mesmo local***	Apoio à Tomada de Decisões Reuniões Eletrônicas Face-a-Face **Edição Cooperativa**	Gerência de Projetos **Edição Cooperativa**
	Local diferente	Video-Conferência Ensino a Distância Teleconferência em Tempo Real **Edição Cooperativa**	Correio Eletrônico Gestão Automática do Fluxo do Trabalho **Edição Cooperativa**

Além da matriz da tabela 2.1, pode-se ainda classificar os *groupwares* segundo seus aspectos funcionais em diversas categorias. Algumas destas são [DIE 96]:

- **Sistemas de Mensagens**: Exemplos mais comuns de groupwares;
- **Sistemas de Suporte a Decisão (GDSS) e Salas de Reunião Eletrônicas**: Implementam, principalmente, a geração e estruturação de idéias, junto com mecanismos de votação;
- **Sistemas de Conferência Eletrônica**: Destaca-se vídeo e áudio conferências;
- ***Sistemas de Workflow***: Apóiam, automatizam ou controlam o processo de trabalho, com grande preocupação no roteamento dos documentos, na forma de formulários eletrônicos, dentro da organização;
- **Sistemas de Autoria Cooperativa**: permitem a um grupo de co-autores compor um documento (ou desenho, projeto, etc.) conjuntamente, utilizando um ambiente compartilhado.

2.3 Ferramentas

Dentre os produtos de CSCW inclui-se sistemas de mensagens, de conferência e videoconferência, sistemas de reunião, de co-autoria e *workflow*. Os sistemas de mensagens como *e-mail* e o EDI - *Eletronic Document Interchange* são os mais antigos e mais importantes tipos de *groupware*. Outros sistemas simples e antigos são os sistemas de conferência textuais, como os *Newsgroups*, onde cada conferência é formada por um conjunto de mensagens e usuários organizados em torno de um (ou

mais) tópico [SOU 96]. A seguir são apresentados algumas ferramentas de *groupware*, cada um com uma breve descrição de suas principais características.

2.3.1 Alliance

O *Alliance* é um editor cooperativo, que permite a usuários distantes, conectados pela Internet, editar de forma assíncrona documentos estruturados.

O suporte a documentos estruturados é ponto importante dentro do *Alliance*. Este suporte baseia-se na API (*Application Programming Interface*) THOT e no editor GRIF.

2.3.2 Framework BW

Um framework consiste em uma mini-arquitetura que fornece uma estrutura geral e o comportamento para uma família de abstrações de software. Um *framework* não é uma aplicação completa, mas aplicações podem ser construídas sobre um ou mais *framework*s. Quando se utiliza um *framework* normalmente só se implementam algumas funções ou se especializam algumas classes, através dos "*hot spot*s", implementados comumente através de *callback*s, polimorfismo ou delegação, permitindo assim que o *framework* seja estendido, adaptado e até combinado com outros *frameworks* [APP 2002].

O *Framework BW (Big Watcher)* foi desenvolvido pela Manuele Kirsch Pinheiro na sua dissertação de mestrado em fev.2001. Consiste em um mecanismo para suporte à percepção de eventos no passado, um mecanismo flexível destinado à obtenção de uma contextualização do ambiente de trabalho em grupo suportado por computador. Este mecanismo foi construído na forma de um *framework,* projetado para ser flexível a ponto de poder ser incluído em qualquer ferramenta de *groupware*, desde que seu autor o queira. Este *framewor*k, chamado de BW *(Big Watche*r), foi organizado em quatro pacotes: três independentes, que trocam informações, descritas no quarto pacote, através somente de classes de fachada. Estas informações são essencialmente eventos, os quais representam as atividades realizadas e já concluídas por algum membro desempenhando um papel dentro do grupo. Estas atividades são registradas pelo *groupware* junto ao *framework,* de modo que este *groupware* possa, através do *framework*, contextualizar seus membros. Além disso, o *groupware* também pode especializar várias classes dentro do *framework* BW, como a descrição dos papéis e a própria descrição dos eventos. Assim, este *framework* pode ser integrado a qualquer ferramenta de *groupware* em ambiente assíncrono que necessite de um mecanismo para o suporte à percepção de

eventos no passado, para evitar que situações de ausência prejudiquem o andamento dos trabalhos [KIR 2001].

2.3.3 Madeus

MADEUS é uma ferramenta de autoria e apresentação de documentos multimídias interativos, parte integrante do projeto CEMT, que visa investigar técnicas de geração de gerênciadores de ambientes de hiperdocumentos de propósito geral, sejam eles de engenharia, software, automação de escritórios ou outra aplicação que lida com coleções de documentos estruturados. É um editor e *player* multimídia, que possibilita a edição cooperativa de forma síncrona e assíncrona de documentos, com uma linha de tempo capaz de organizar e sincronizar a apresentação de diferentes documentos multimídias.

2.3.4 Thot

O THOT é um sistema genérico para o desenvolvimento de aplicações baseadas no conceito de documentos ativos estruturados. Ele fornece um conjunto de funções de manipulação de documentos, com sua API, e permite que outras aplicações incluam funções extras pelo mecanismo de *callback*. Além disto, o THOT também permite incluir uma linguagem declarativa simples, para gerar interfaces do usuário de uma aplicação.

Um documento THOT é representado por sua estrutura lógica, chamada Estrutura de Árvore Abstrata. Esta é uma estrutura principalmente hierárquica, com relações suplementares, representando as ligações não-hierárquicas, formando um hipertexto. Com o uso dessa estrutura hierárquica, o THOT determina uma estrutura genérica, uma espécie de tipo para o documento. Este tipo objetiva ajudar, ou obrigar, quando necessário, o usuário a produzir um documento de acordo com a estrutura genérica. A estrutura da Árvore Abstrata segue algumas regras, que são definidas em um esquema de estrutura. O aspecto gráfico dos documentos, de acordo com o “tipo”, também é especificado por um esquema de apresentação. O THOT permite que vários esquemas de apresentação sejam associados a uma estrutura genérica. Estes esquemas vão permitir que o aspecto gráfico do documento seja modificado globalmente, apenas alterando-se o esquema de apresentação em uso. A presença de vários esquemas vai atuar como visões diferentes do documento.

Além da API THOT, existe também um editor estruturado homônimo, construído sobre a API, fazendo uso intenso de todas as suas características.

2.3.5 WebDAV

O *Web*DAV "*Distributed Authoring and Versioning on the Web*" foi desenvolvido por um grupo de trabalho formado pela IETF – *Internet Engeneering Task Force*, e pelo W3C – *World Wide Web Consortium*, que trata sobre a redação distribuída e a gerência de versões sobre a WWW. Seu principal objetivo consiste em estender o protocolo HTTP, a fim de que este possa suportar características de autoria e controle de versões e com isso fornecer uma arquitetura aberta ao nível de protocolo para o desenvolvimento de aplicações para a cooperação assíncrona de documentos na *Web*, tanto em formato HTTP, quanto XML.

Para tanto, pontos como controle de acesso, *lock*, mecanismos de autenticação, criação, modificação, redirecionamento e recuperação de recursos e gerência de versões, incluindo mecanismos de *check-in / check-out*, fusão automática e acesso a versões anteriores deverão ser abordados. E embora este seja um grupo recente, algumas contribuições, juntamente com as linhas gerais do grupo podem ser encontradas em [INT 2002].

2.4 Cooperação

De acordo com [GRI 2000] os sistemas de gerência de *workflow* tradicionais não estão adaptados para a execução de processos cooperativos, não suportam as numerosas mudanças que podem ser necessárias entre os usuários para a execução destas tarefas.

Assim [GRI 2000] propõe o uso combinado de um protocolo de transações cooperativas com um modelo tradicional de *workflow*, também encontrada em [GOD 2000]. Esta proposta vem ao encontro à proposta deste trabalho, sendo este o motivo pelo qual esta proposta foi tomada como base para a proposta de cooperação do presente trabalho, que combinada com o a idéia do Tickertape e outras funcionalidades propostas, atinge os objetivos desta dissertação.

A proposta de [GRI 2000] é permitir a troca de dados durante a execução das tarefas, antes de conclui-las e ainda permitindo a antecipação de algumas tarefas em pontos pré-determinados, utilizando os dados das tarefas que ainda não concluíram, mantendo a consistência do sistema através de um protocolo de regras que determinam como devem ser utilizados os dados intermediários e como gerênciar o processo com tal característica. O protocolo ao qual o trabalho de [GRI 2000] se refere é apresentado em [GOD 99], o qual apresenta uma proposta de um novo operador, a ser utilizado na representação gráfica do *workflow* indicando a presença de trabalho cooperativo. Sendo a proposta composta pelo operador na parte gráfica e um conjunto de regras na implementação da gerência do *workflow* estas regras permitem a utilização de resultados intermediários de tarefas ainda não concluídas, mas que segundo a utilização desta informação pode-se possibilitar que dois ou mais participantes trabalhem cooperativamente na execução de uma tarefa, como também propiciar a execução

antecipada de uma tarefa. Este operador com suas regras não cobre todos os aspectos de cooperação e coordenação possíveis, pois está dedicada principalmente ao gerênciamento de interações cooperativas baseadas na troca de documentos.

Na figura 2.1 abaixo temos a notação gráfica proposta por [GOD 99] para o seu operador (COO) e o conector (And-Join) normalmente utilizado nas representações de workflow e na figura 2.2 um exemplo de utilização onde o operador (COO) é utilizado para informar que as atividades “Fazer Exercício” e “Corrigir Exercícios” são realizadas de forma cooperativa e que após a conclusão destas duas tarefas (conector And-Join) é realizada a tarefa de “Atribuir as notas”.

FIGURA 2.1 Operador “COO” e conector “And-Join”

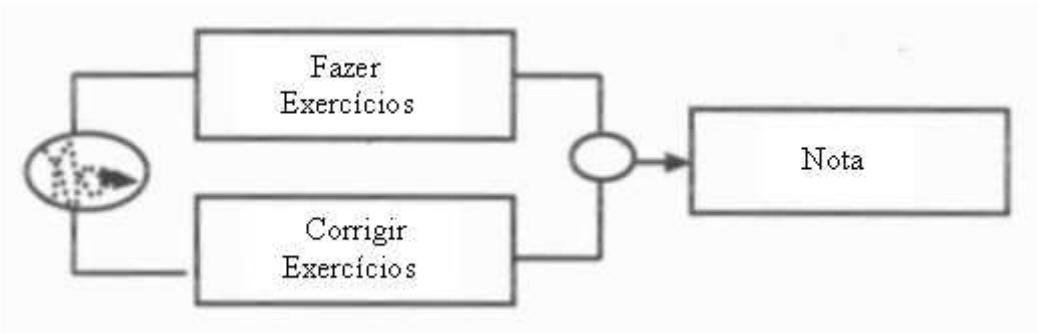


FIGURA 2.2 Exemplo de utilização do Operador “COO”

As regras que compõe este operador (COO) foram definidas com o objetivo de manter a integridade das informações utilizadas na execução das tarefas, mesmo com o compartilhamento de resultados intermediários, através da troca de documentos existentes em um repositório [GOD 2001]. Assim [GOD 99] apresenta 10 regras para o seu protocolo:

1. Um resultado produzido antes do fim de um processo é sempre um resultado intermediário. Usuários podem chamar a qualquer tempo a operação *IR-write* para produzir um resultado intermediário.
2. Um resultado produzido no fim de um processo é um resultado final. Todos os resultados finais são produzidos atomicamente durante a execução da operação *terminate*.
3. Um processo que produz um resultado intermediário deve produzir um correspondente resultado final. O protocolo coleta todos os objetos que foram *IR-written* pelo processo e automaticamente produz um resultado final para cada um deles durante a fase de conclusão do processo.
4. Se um processo ler um resultado intermediário, então ele deve ler o correspondente resultado final. O sistema mantém relações de dependência entre os processo spara memorizar o fato de que um processo leu um resultado intermediário de outro processo. Quando uma atividade A ler um resultado intermediário de um processo B, é criada uma dependência entre as atividades A

e B, quando a atividade A ler o resultado final do processo B, a dependência que existia será eliminada.

5. Um processo não pode terminar se ele ainda está dependente de um outro processo. Se um processo tentar terminar sem ler o resultado final ao qual está dependente a operação *terminate* será abortada e o processo continuara ativo.
6. Processos envolvidos em dependências cíclicas formarão um grupo de processos.
7. A sendo membro de um grupo de processo pode iniciar a operação de conclusão do grupo, pela tentativa de terminar a si próprio. A operação de neste caso produz um conjunto de potenciais resultados finais e muda o estado do processo de ativo para “pronto para terminar” (RTT).
8. Quando um membro de um grupo tenta concluir e todos os outros membros do grupo estão no estado pronto para terminar, então todos os processos são concluídos simultaneamente. Os potenciais resultados finais são definitivamente promovidos para resultados finais.
9. Quando um membro do grupo tenta concluir e existe ainda algum membro ativo, então isto produz novos potenciais resultados finais e passa para o estado “pronto para terminar”.
10. Se um membro do grupo produz um novo resultado intermediário durante a fase de conclusão do grupo, então esta tentativa de conclusão do grupo é abortada, e todos os membros do grupo voltam ao estado ativo. Esta é a forma para uma atividade claramente indicar que está em desacordo com o resultado produzido pelo grupo e questionar por mais trabalho nos objetos compartilhados.

Com este conjunto de regras e o operador gráfico, [GOD 99] propõe viabilizar a cooperação na execução das tarefas pela troca de documentos presentes em um repositório coletivo de acesso comum a todos os participantes do processo[GOD 2001], sendo esta uma proposta interessante e útil para alcançar os objetivos propostos neste trabalho, a proposta de [GOD 99] foi tomada como modelo para a implementação de cooperação.

Outra proposta interessante e que também foi tomada como modelo e utilizada para melhorar a cooperação na execução das tarefas no presente trabalho é a proposta de [PAR 98], no qual ele propõe o Tickertape.

Baseando-se na idéia de que cooperação advém da troca de informações entre os participantes de um processo, uma ferramenta que possibilite esta comunicação em um grupo que efetivamente utilize esta informação para cooperar na execução das tarefas, pode ser considerada como fomentadora da cooperação do grupo. Assim o Tickertape nasceu quase acidentalmente, pois inicialmente foi desenvolvido para a troca de informações gerais entre as pessoas, cada um dando-lhe a utilização que lhe convinha, sendo um canal aberto de comunicação onde é possível criar um *profile*, uma configuração individual onde cada um filtra o que lhe convém, e transmite o que deseja. Com a sua utilização percebeu-se o seu potencial para viabilizar a cooperação, passando assim a ser estudado e desenvolvido com este objetivo.

O Tickertape é uma ferramenta de mensagens, que tem como uma de suas vantagens o fato de ser pequena, apenas 1 linha, não poluindo o ambiente de trabalho,

podendo ficar visível o tempo todo. Nesta única linha são apresentadas as mensagens a medida que são enviadas, mas permite também a rolagem das mensagens de forma que se possa ler qualquer mensagem enviada a qualquer tempo. O Tickertape assemelha-se a um chat, onde vemos apenas uma linha por vez e podemos definir o que queremos ver, o tipo de mensagem, quais os remetentes, e ainda podemos mandar mensagens da mesma forma, criar grupos, etc.

Abaixo na figura 2.3 vemos o Tickertape mostrando uma mensagem e na figura 2.4 a interface de envio de mensagens.

FIGURA 2.3 Mensagens no Tickertape

FIGURA 2.4 Enviando mensagens com o Tickertape

Outro aspecto que pode ser considerado na colaboração refere-se aos tipos de comportamento cooperativo. Em [LAB 98] temos uma apresentação de 4 comportamentos cooperativos. [LAB 98] apresenta o desenvolvimento de um ambiente cooperativo o WEICOT, o qual contem uma definição de tarefas composta por 4 componentes: Características Intrisecas; Restrições, Cooperação e o Corpo da tarefa. No componente COOPERAÇÃO encontramos 4 propriedades: Escopo; Alocação; Comportamento Cooperativo e Capacidade de Delegação, sendo a cooperação promovida pelo gerênciamento destas informações. O WEICOT considera que a cooperação é resultado da adoção de uma atitude (um comportamento cooperativo) compartilhado por um conjunto de agentes que executam uma tarefa. Esta atitude reflete um comportamento social que pode caracterizar o grupo, distinguindo então os seguintes modos:

- Competição: A mesma tarefa é alocada para todos os agentes. Cada um tenta fazer a tarefa separadamente antes ou no lugar dos outros. Este modo de trabalha não implica necessariamente no desconforto entre os agentes, Porém utiliza mal os recursos.

- Co-ação: Os agentes realizam a tarefa separadamente, mas sem competição. Não há conflitos na utilização dos recursos. Porém é necessário definir algum critério de escolha entre as soluções apresentadas.
- Assistência: A tarefa é alocada para somente um agente no grupo. Os outros podem ajudar este agente se for necessário.
- Negociação: Os agentes negociam a solução do problema. Se um consenso não é alcançado um meta-agente pode intervir neste caso para definir a solução.

Se for necessário outros comportamentos cooperativos podem ser inseridos, conforme a necessidade do sistema. O importante é através destes recursos proporcionar aos agentes um ambiente onde possam interagir e compartilhar informações suficientes e de forma eficiente para obterem um incremento sinérgico na execução das tarefas.

2.5 Projeto CEMT

O projeto CEMT "*Conception of Cooperative Environment for Editing Multimedia Documents with Workflow Technology"* [OPE 2001] [TEL 2001], financiado pela cooperação CNPq-INRIA, tem como um dos parceiros a UFRGS, sendo o prof. José Valdeni de Lima o coordenador no Brasil. O projeto tem como foco a autoria cooperativa de material para cursos à distância, através da construção de um ambiente cooperativo guiado por *workflow*. Desta forma este trabalho se enquadra nos objetivos do projeto, tendo em vista que o objetivo deste trabalho é a obtenção de uma estrutura de *workflow* que suporte percepção de eventos. O ambiente proposto pelo projeto CEMT tem sido todo baseado em ferramentas e padrões abertos, mas procura também a aplicação de padrões aceitos no mundo, segundo os organismos reguladores de cada segmento como W3C e WfMC (*Workflow Management Coalition*), objetivando também a interoperabilidade compatibilizando-se com as novas tecnologias, como o SMIL *(Synchronized Mutimedia Integration Language).* O projeto CEMT tem concentrado esforços no uso de padrões WEB, sendo esta uma rede que interliga as pessoas de todo o mundo eliminando as distâncias que as separam. Com o auxilio de ferramentas que propiciem o ambiente adequado a WEB pode ser utilizada tanto pelos alunos para fazerem seus cursos a distância on-line ou off-line, quanto pelos mentores e idealizadores dos cursos para também de forma individual ou cooperativa desenvolverem seus cursos on-line ou off-line, falta apenas o desenvolvimento de tais ferramentas que possibilitem extrair da WEB o máximo do seu potencial, para cooperação e ensino.

A principal ferramenta utilizada para autoria de EAD no contexto do projeto CEMT é o editor Amaya um *open source* desenvolvido pela W3C, que por ser o ambiente de teste do W3C, inclui os mais atuais padrões da WEB, como *XML, SVG* e *CSS* (*Cascading Style Sheets*), além de ser um editor *WYSIWYG (What Uou Get Is What I Get),* é facilmente expansível.

2.6 WfMC

Fundada em agosto de 1993, a WfMC é uma organização internacional sem fins lucrativos formada por fabricantes, usuários, analistas e pesquisadores da tecnologia de *workflow*. Sua missão é promover e desenvolver o uso de *workflow* através do estabelecimento de padrões de terminologia, interoperabilidade e conectividade entre os produtos de *workflow* [XML 2003e].

A WfMC estabeleceu vários grupos de trabalho, cada um em uma área particular de especificação. Os grupos de trabalho são estruturados livremente em torno do "Modelo de Referência de *Workflow*" que provê o ambiente para o programa de padrões da WfMC. O Modelo de Referência identifica as características comuns de sistemas de *workflow* e define 5 interfaces funcionais discretas pelas quais um sistema de administração de *workflow* interage com seu ambiente [WOR 99].

Muitos dos produtos de *workflow* que existem atualmente foram desenvolvidos para suplementar algumas necessidades existentes em produtos de outras áreas, como as de sistemas de imagens, de e-mail ou de *groupware*. Características rudimentares de *workflow*, tais como o roteamento, foram adicionadas a estes sistemas do modo a que convinha a cada fabricante, gerando ferramentas com conceitos completamente diferentes [BRE 2001]. Muitos conceitos, existentes em um modelo, são representados de forma completamente diferentes em outros, ou chegam até mesmo a não existir. Os prejuízos inerentes a estas diferenças e a falta de um modelo conceitual amplamente aceito e utilizado são:

- A impossibilidade de se modelar um processo de *workflow* independente de implementação, que possa ser criado numa ferramenta, por exemplo, e utilizado em outra. Isto acarreta um problema de interoperabilidade, que vem sendo atacado pela WfMC através de uma série de padrões.
- A inexistência de um modelo que possua todas as características desejadas em um modelo de *workflow*. Cada modelo possui, de acordo com a sua origem, as suas peculiaridades, que ressaltam determinados conceitos e ignoram outros. Infelizmente, não existe ainda um modelo reconhecido que tenha incorporado todos os pontos fortes dos demais modelos. Como conseqüência, o mapeamento de um processo de negócios em um determinado modelo pode requerer o uso de adaptações para expressar conceitos não incorporados pelo modelo. Tais adaptações, em ultima instância, originam a perda de informações sobre a realidade modelada.
- As diferentes formas que um conceito é expresso entre os diversos modelos dificultam a compreensão do processo de negócio, uma vez que, para alguns usuários ou analistas, uma determinada expressão ou símbolo pode representar conceitos completamente diferentes em modelos distintos.

2.7 Considerações Finais

As tecnologias apresentadas foram desenvolvidas para suprir necessidades especificas, porém, de forma individual, necessidades estas que se tem evoluído ao longo do tempo, de acordo com a evolução da humanidade. Hoje necessitamos cada vez mais de sistemas capazes de compartilhar informação, de forma síncrona e assíncrona e que possam gerênciar o fluxo e o cronograma de trabalho. É justamente neste ponto em que entram em cena os sistemas de *workflow* gerênciando o fluxo e o cronograma de trabalho, mas ainda lhe falta a percepção e o compartilhamento de informações. O próximo capítulo detalha o aspecto da percepção.

3 Awareness

Awareness “Percepção” segundo [SOH 98] é um estado da mente, e não pode ser criado por sistemas. Os sistemas apenas podem estimular o processo cognitivo necessário, para que o usuário tenha a percepção do ambiente e dos demais integrantes. A aplicação de *Awareness* a sistemas de *workflow* é o foco do presente trabalho, necessário para que se otimize ao máximo o trabalho de um grupo, sendo ele informatizado ou não. Principalmente quando se está desenvolvendo um trabalho em grupo, auxiliado por computadores e seus respectivos programas, pois neste ambiente informatizado com os colaboradores próximos ou não, desenvolvendo suas atividades simultaneamente ou em momentos diferentes, a organização das atividades e a coerência entre estas atividades dependem fundamentalmente da percepção dos eventos, da percepção do “estado” do trabalho, da percepção da cronologia do desenvolvimento das atividades.

3.1 Percepção

Percepção. O que é percepção? Perceber? Perceber o que?

A percepção a que este trabalho se refere é a percepção das atividades passadas, presentes e futuras integrantes de um projeto, o qual foi especificado em um *workflow*.

Esta percepção diz respeito aos integrantes, quem são, o que estão fazendo, o que fizeram, o que deverão fazer, quem já concluiu sua tarefa, quem está atrasado, quem deverá fazer a próxima tarefa, o que deverá ser feito, por quem? Onde? Quando? O porque das situações, enfim, tudo o que se deva saber para que o trabalho seja coerente de maneira que a contribuição de cada integrante seja somatória à participação dos demais, e assim se alcance o resultado de um trabalho de grupo bem coeso e organizado.

Generalizando, pode-se afirmar que *"awareness* significa uma compreensão do estado total do sistema, incluindo atividades passadas, *status* atual e opções futuras" [SOH 98]

3.1.1 Analisando os Questionamentos

Estes questionamentos vistos no parágrafo anterior representam a “percepção” de uma forma resumida, assim vamos analisar cada item:

- **O Que** – Refere-se ao objeto, tarefa, integrante ou uma informação qualquer que se deseja obter naquele momento, assim a resposta ao O que? Embora esteja dentro de determinadas possibilidades, depende do

contexto, do objetivo, de quem está perguntando, do ambiente síncrono ou assíncrono dessa forma pode ser o que estão fazendo, o que há para fazer, o que está atrasado, quem está com a tarefa em atraso, quem já concluiu, etc.

- **Quando** – Percepção de tempo, de temporalidade, de ordem cronológica, refere-se a dois aspectos essenciais: quando ocorreram os eventos geradores das informações de percepção e quando se dá a apresentação destas informações aos usuários (quando fornecer a percepção) [KIR 2001]. Outra característica importante com relação ao questionamento "Quando" é quando apresentar as informações. Em sistemas síncronos, quanto mais cedo for dada a percepção destes eventos ao grupo, melhor (imediato), já em sistemas assíncronos a informação da percepção, além de ser persistente, é apresentada aos usuários possivelmente em um momento posterior à ocorrência dos eventos, já que não há garantias quanto a este intervalo de tempo entre a ocorrência e a percepção pelos usuários.

- **Onde** – Se refere a onde as informações são geradas e apresentadas, o espaço de trabalho e os objetos compartilhados, onde o trabalho deve ser executado, montando assim de forma virtual o que seria o ambiente físico de trabalho. Assim pode ser possível a percepção do ambiente e do trabalho em grupo pelos integrantes como se estivessem realmente todos no mesmo ambiente físico e ao mesmo tempo. Assim esta percepção do que está ocorrendo neste espaço de trabalho compartilhado é chamada de *workspace awareness* e pode ser comparada à percepção que as pessoas ao redor de uma mesa em uma reunião têm umas das outras e do trabalho que está sendo feito [KIR 2001].

- **Como** – Refere-se a de que forma a informação é mostrada ao usuário pela interface, de que maneira a interface interage com os usuários. A interface deve apresentar resumidamente a informação de forma natural e espontânea surgindo diretamente das atividades de cada pessoa. De acordo com o ambiente, WYSIWIS _ *What You See Is What I Se*e, e WYSIWIS relaxadas, síncronos ou assíncronos, a interface terá maior ou menor acoplamento e a questão "Como" terá diferentes características.

- **Quem** – Refere-se a quem está trabalhando neste momento, os participantes do grupo em um determinado instante. Este é um tipo de informação muito comum em um trabalho de grupo real, onde com apenas um olhar todos sabem quem está trabalhando no grupo, porém no caso de sistema informatizados, deve existir esta informação disponível ao usuário, fornecida pela interface, pois não há outra maneira pela qual os participantes possam saber quem está trabalhando no grupo em dado momento. Desta forma a noção de presença será obrigatória em alguns momentos e em outras apenas deve existir a oportunidade, a disponibilidade da informação. A figura 3.1 abaixo, retirada de [GRE 2002], mostra um exemplo de mecanismo implementando pelo *GroupKit* [ROS 96], [ROS 97], que apresenta uma lista com os participantes

presentes e ainda permite ao usuário consultar maiores informações sobre os colegas presentes, como foto, telefone e correio eletrônico.

FIGURA 3.1 Informações Detalhadas Sobre um Participante

- **Quanto** – Refere-se a quanto de informação é suficiente, isto por que o usuário deve receber informação suficiente para a percepção do grupo e do trabalho do grupo, porém não em excesso a ponto de sobrecarrega-lo. A questão “Quanto” esta relacionada a todos os demais questionamentos da percepção, desta forma o critério de “Quanto” deve ser relacionado a soma de todas as informações advindas da percepção, posicionando-se de maneira coerente em um ponto médio entre a insuficiência e a sobrecarga de informações, como pode ser representado pela figura 4.1 abaixo:

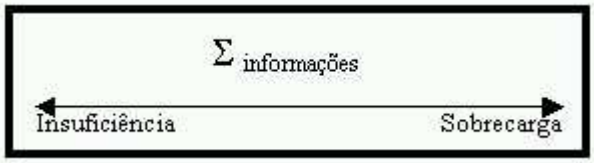


FIGURA 4.1 A dimensão “Quanto”

3.2 Considerações Finais

Como foi visto neste capítulo, a percepção pode ser traduzida como o conhecimento do estado atual do sistema, o andamento dos trabalhos, a distribuição das tarefas, o histórico da execução de uma ou de todas as atividades. Dentro deste contexto o presente trabalho aborda estas questões no capítulo 6, no desenvolvimento do modelo, almejando fornecer uma noção de consciência do sistema como um todo, a cada um dos usuários, proporcionando-lhes um ambiente de trabalho cognitivo e apto a atender as suas necessidades.

4 WorkFlow

Normalmente o trabalho é organizado pelo indivíduo mais capacitado para tal função, pelo seu conhecimento e experiência. Este por sua vez determina quem executa cada atividade, escolhendo aqueles que tem mais habilidade, conhecimento ou experiência para executar a função. Algumas ferramentas podem ser utilizadas para auxiliar na execução das tarefas, mas toda a parte de gerência, organização e revisão do trabalho, é de responsabilidade do coordenador. Porém nos últimos anos tem-se desenvolvido a área de gerência e modelagem de *workflow*, permitindo não apenas a representação do processo, mas também o seu gerênciamento por sistemas automatizados.

Um esquema de *workflow* representa uma abstração de um processo de negócio, sendo modelado e posteriormente executado [ZSC 2002]. *Workflow* pode ser definido como sendo um conjunto de atividades, que podem, ou não, serem executadas simultaneamente, com alguma especificação de controle e fluxo de dados entre estas atividades, podendo representar vários processos e seus relacionamentos. Outra definição que pode ser dada é a automatização de um processo, em todo ou em parte, durante o qual são passados documentos, informações ou tarefas de um participante para outro, de acordo com um conjunto de regras processuais [FIS 2001].

Sistemas de *workflow* estão se tornando, cada dia mais, líderes de mercado na modelagem de regras de negócios das organizações [BRE 2001], estes sistemas de *workflow* consistem de componentes de modelagem e componentes de representação do sistema.

E com o surgimento de sistemas de *workflow*, se fez necessário o desenvolvimento de sistemas de administração de *workflow*, os quais *são* sistemas que definem, criam e administram a execução de *workflow* através do uso de *software*, rodando em uma ou mais máquinas de *workflow*, com a qual se pode interpretar a definição de processo, interagindo com os participantes do *workflow* e, onde necessário, invoque o uso de ferramentas e aplicações [FIS 2001].

Os modelos precisam disponibilizar expressões relacionadas aos processos, restrições temporais, trocas dinâmicas e tratamento de exceções, o que nem sempre acontece [BRE 2001]. O sistema de *workflow* precisa habilitar execução dinâmica, modelagem baseada em processos, providenciar coordenação e assistência aos usuários a cada passo da modelagem, e processamento distribuído. Existe ainda a necessidade dos sistemas de *workflow* manipularem e monitorarem a informação relativa ao fluxo de trabalho para gerênciar, coordenar e controlar o trabalho mais eficientemente e para proporcionar recuperação histórica do fluxo de trabalho permitindo assim monitoramento histórico (auditorias).

Embora existam diversos modelos como Modelo Casati/Ceri [CEP 2001] [ZSC 2001], Modelo de Gatilhos [JUN 2000] [CEP 2001], Método Cimosa de Modelagem de Negócios [CEP 2001], TF-ORM [SIZ 99], Modelo de Ações [JUN 2000] e Modelo de Referência da WfMC [JUN 2000] [SIZ 99], Modelo Baseado em Comunicação [SIZ 99], Modelo baseado em Atividades [NIC 98], ainda assim não contemplam todos os requisitos para percepção de eventos durante a execução [NIC 99].

O *Workflow* tem como um dos seus principais objetivos minimizar o problema da coordenação do trabalho nos processos de negócios. Desde que o processo de *workflow* represente os aspectos dinâmicos e estáticos do processo, é possível inferir que uma pessoa pode identificar os valores dos principais aspectos estruturais de uma

organização através da análise do processo do *workflow* [IOC 2002]. O principal problema das atividades baseadas em processos é que não se tem o controle total das atividades que devem ser executadas. Além disso, não se têm uma visão de quais informações estão sendo manipuladas por essas atividades em um dado instante e nem quem as manipula. As técnicas de modelagem do *workflow* devem ter como objetivo básico, minimizar estes problemas.

A utilização e a automação de sistemas de workflow são importantes na medida em que proporcionam às organizações uma melhor coordenação e distribuição de responsabilidades entre os participantes de um processo [JUN 2000].

Os sistemas de *workflow* quando desenvolvidos de modo a apresentarem as capacidades de percepção de eventos e o compartilhamento das informações de processos concluídos e de processos em execução, permitem as organizações otimizar recursos e processos, alem de obter-se um incremento na eficiência da organização e redução no tempo necessário para a realização de um processo. Estas características somadas ao fato de se ter o processo modelado, pode com o auxilio de um bom algoritmo de gerência do processo e tratamento de exceções, reduzir a necessidade de chefias, tornando a estrutura organizacional da empresa mais horizontal e ainda a documentação dos processos de tal forma que pode eliminar a ocorrência de pessoas insubstituíveis ou de problemas nas trocas de pessoal e ainda facilita o rodízio de funções dentro da organização. Fatores estes que aumentam a eficiência da organização, tornando-a mais competitiva e com maiores possibilidades de alcançar suas metas e ainda com maior longevidade, em relação ao que seria obtido sem o uso desta tecnologia.

Assim é de grande importância o desenvolvimento de tais sistemas de *workflow*, sendo este trabalho um passo a mais em direção a este objetivo.

4.1 Definições Relacionadas a Workflow

Vejamos agora algumas definições relacionadas a *workflow* extraídas do glossário da WfMC [WOR 96].

- **Aplicação Chamada**: Aplicação de *workflow* que é invocada pelo sistema gerênciador de *workflow* para automatizar uma atividade ou ajudar um participante de workflow no processamento de um item de trabalho.
- **Atividade**: Uma descrição de um fragmento de trabalho que contribui para o cumprimento de um processo, associada a um intervalo de tempo, pode ser automatizada ou manual.
- **Ator ou Participante**: Recurso (humano ou automatizado) que executa o trabalho representado por uma instância de atividade de um *workflow*.
- **Evento**: Algo que ocorra em um instante de tempo especifico.
- **Gatilho (*Trigger*)**: Disparo de uma atividade por um evento, podendo ser visto como uma regra que é avaliada em função da ocorrência de um evento.
- **Instância (de um processo ou atividade)**: A representação de uma única execução de um processo, ou atividade dentro de um processo, incluindo os seus dados associados. Cada instância representa uma

diferente linha de execução do processo ou atividade, a qual pode ser controlada independentemente e terá o seu próprio estado interno e identidade externamente visível.

- **Item de Trabalho**: Representação do trabalho a ser processado por um ator.
- **Lista de trabalho**: Lista de itens de trabalho associada a um determinado ator.
- **Objeto**: Alguma coisa que seja capaz de ser vista, tocada ou sentida.
- **Papel**: Conjunto de atores que apresentam um conjunto especifico de atributos, qualificações e/ou habilidades (características) que os tornam aptos a executarem a atividade relacionada ao papel.
- **Processo**: É uma representação de um processo de negócio através de uma rede de atividades e seus relacionamentos, critérios para indicar o início e o término do processo, e informações sobre atividades individuais, tais como participantes, dados, etc.
- **Processo de Negócio**: Um conjunto de um ou mais procedimentos ou atividades relacionados os quais realizam coletivamente um objetivo de negócio ou meta, geralmente dentro do contexto de uma estrutura organizacional que define papéis funcionais e relacionamentos.
- **Sincronismo**: Representação formal da interação das atividades através do fluxo de controle de *workflow*.
- **Sistemas de *Workflow***: Um sistema de *workflow* contém um *workflow*, todos os atores, todas as estruturas e o significado envolvendo o *workflow*.
- ***Workflow***: Sistema cujos elementos são atividades, interagindo umas com as outras através de triggers e disparadas por eventos externos.

4.1.1 Classificação dos Tipos de Atividade

Podemos classificar as atividades existentes na especificação de um *workflow* em:

Atividade Baseada na Instância do Workflow x Atividades Batch**: Atividades que lidam exclusivamente com uma única instância do workflow são atividades baseadas na instância do workflow. Atividade batch é aquela que só pode ser completada se um conjunto de instâncias de um determinado workflow estiverem sendo executadas em conjunto, e o produto da atividade for adicionado a um sub-conjunto das instâncias.**

- **Estruturada x Não Estruturada**: Atividades estruturadas são aquelas bem entendidas pelo sistema e pelas pessoas que especificam o procedimento, enquanto que atividades não estruturadas são aquelas que não podem ser colocadas sob supervisão do sistema.
- **Externas x Internas**: Externas são as atividades executadas fora do sistema de *workflow* e internas são as atividades executadas dentro do sistema de *workflow*, e monitoradas pelo mesmo.
- **Quanto a Finalidade (Função):**
  - **Busca de Dados Externos (*External Data Gathering*)**: Obtenção e armazenamento de dados de fontes externas

- **Processamento de Dados (*Data Processing*)**: Derivam novas informações a partir de dados externos, adotando critérios internos.
- **Decisão (*Decision*)**: Envolvem a escolha entre alternativas através de um julgamento subjetivo. Este tipo de atividade determina qual o caminho que o fluxo irá seguir.
- **Autorização (*Authorization*)**: Forma fraca de decisão, onde uma pessoa da organização assume a responsabilidade pelo estado da instância do *workflow*
- **Espera (*Waiting*)**: A instância do *workflow* fica esperando por algum *trigger* para dar seguimento à próxima atividade.
- ***Schedule Temporal (Temporal Scheduling)***: Determinam um momento ou a ordem na qual um conjunto de atividades será executado.
- **Atividades *Proxy* (*Proxy Activities*)**: Tipo de atividade de espera que só será executada após a ocorrência de um número de execuções de uma atividade externa.
- **Alocação (*Axtor Assignment*)**: Determina que executará a instância de uma outra atividade.

Relacionada ao Atributo Tempo:

- **Tempo de Execução (Duração)**: Representa quanto tempo a atividade levará para ser executada sob condições normais
  - **Determinado**: Sabe-se previamente quanto tempo a atividade levará para ser executada.
    - **Exato**: Atividade possui um tempo exato de execução.
    - **Não-Exato**: Não se conhece o tempo exato de execução, mas sim os tempos mínimo e máximo de execução.
  - **Indeterminado**: Impossível, num primeiro instante, definir o tempo de execução.
- **Data e Hora de Inicio:** Indica que a atividade deve ter seu início na data e hora pré-estabelecidas.
  - **Dependente do Tempo**: Possui data e hora pré-estabelecidos
  - **Independente do Tempo**: Não possui data e hora pré-estabelecidos.
- **Data e Hora de Término:** Indica que a atividade deve ter seu término até a data e hora pré-estabelecidas.
  - **Dependente do Tempo**: Possui data e hora pré-estabelecidos
  - **Independente do Tempo**: Não possui data e hora pré-estabelecidos.

4.2 Características e Funcionalidades Comuns aos Sistemas de *Workflow*

Apesar da sua grande diversidade, os sistemas de *workflow* apresentam um certo conjunto de características e funcionalidades comuns. As principais são:

- **Roteamento de Trabalho**: Possibilita a predefinição da seqüência em que as atividades serão executadas. Tipicamente, o participante do *workflow* recebe um

item de trabalho e, quando termina o seu processamento, habilita a atividade seguinte para que ela seja iniciada. O roteamento pode ser seqüencial ou baseado numa decisão. No segundo caso, um teste, ou uma regra, é testado a fim de verificar quais atividades devem ser disparadas.

- **Monitoramento e Controle**: São funções que fornecem informações a respeito da execução de um processo de *workflow*, isto é, acompanham uma determinada instância e descobrem o seu status atual de processamento, o responsável atual por sua execução, e quanto tempo ela está esperando na atividade atual, etc.
- **Notificação**: Com a coordenação do trabalho, a sua distribuição é realizada através de notificações que informam os usuários sobre as tarefas que eles devem realizar e sobre algum eventual estouro do tempo. Este estouro pode ocorrer quando ela possui uma data de término pré-determinada.
- **Distribuição do Trabalho**: Na definição de um processo, geralmente, não é especificado quem será o participante que executará cada tarefa, mas sim o papel a qual cada atividade está atrelada. Este papel, via de regra, encontra-se ligado a diversos participantes da organização.
- **Gerênciamento de Procedimentos**: Um sistema de *workflow* deve oferecer aos usuários finais meios para se redefinir os passos de *workflow*, as seqüências dos seus passos, etc.

4.3 Tipos de Workflow

Vamos agora analisar os tipos de *workflow* definidos na bibliografia consultada, são eles:

4.3.1 Ad Hoc

Workflows do tipo *Ad Hoc* descrevem processos simples, envolvendo um grupo pequeno de profissionais onde é difícil encontrar um esquema para coordenação e cooperação de tarefas. Durante a execução deste tipo de *workflow*, a parte referente à coordenação é controlada por pessoas. As atividades suportadas geralmente requerem soluções rápidas e são executadas apenas uma vez.

Os tipos de tarefas relacionadas ao *workflow* do tipo *Ad Hoc* consistem de procedimentos rotineiros de escritório como aprovações, revisão de documentos, documentação de produtos ou pedidos vendidos, onde não há um padrão de movimentação de informação entre pessoas.

Workflow Ad Hoc ou Colaborativo é sempre usado em áreas administrativas de uma organização, sendo caracterizado pela negociação e a definição de um novo *workflow* para cada uso [FIS 2001].

4.3.2 Administrativo

Nesta categoria, enquadram-se *workflows* onde as atividades são fracamente estruturadas, repetitivas, previsíveis e com regras de coordenação de tarefas simples [FIS 2001]. *Workflows* administrativos não englobam um processo de informações complexas. Nesse caso, a ordenação e coordenação de tarefas podem ser automatizadas.

Aplicações desse tipo geralmente apresentam um processo com a sua estrutura pré-definida por um "administrador de *workflow*", sendo que a complexidade de roteamento é baixa. Tais aplicações têm o objetivo de alcançar um processo mais rápido e eficiente através da coordenação de atividade de vários indivíduos ou grupos.

4.3.3 De Produção

Envolvem processos de negócio repetitivos e previsíveis. Diferentemente de *workflows* administrativos, englobam atividades altamente estruturadas que descrevem processos de informações complexas. Geralmente sua execução requer um número alto de transações acessando múltiplos sistemas de informação. Sistemas de gerênciamento de *workflow* que suportam este tipo de *workflow* devem permitir definição de relações complexas entre tarefas e devem controlar a execução da tarefa com pouca intervenção humana [FIS 2001].

4.3.4 Orientados a Pessoas e a Sistemas

Em *workflows* orientados à pessoas, os agentes são pessoas que cooperam, executam tarefas e garantem a consistência dos resultados do *workflow,* enquanto que em *workflows* orientados a sistemas são os sistemas de computadores que executam operações computacionais intensas e softwares especializados em determinado tipo de tarefa, controlam e coordenam tarefas automatizadas com pequena ou nenhuma intervenção humana. Sendo assim, devem fornecer controle de concorrência e mecanismos de recuperação para garantir a consistência e confiabilidade.

4.3.5 Transacional

Envolve execução e coordenação de diversas tarefas que podem envolver humanos e sistemas autônomos distribuídos e heterogêneos. Devem suportar o uso de

propriedades tipicamente transacionais, como atomicidade, consistência, isolamento e durabilidade permitindo a especialização das atividades de cada *workflow*.

4.3.6 Horizontal vs. Vertical

Outra segmentação da tecnologia de *workflow* é às vezes chamada de *workflow* horizontal vs. vertical. Este tipo de *workflow* move o trabalho através da organização, de pessoa para pessoa ou para diferentes departamentos ou sistemas. Algumas pessoas chamam o *workflow* horizontal de "*routing*". Este é o tipo mais comum de *workflow* [FIS 2001].

4.4 Modelagem de *Workflow*

Sistemas de *workflow* existem há pouco tempo e embora tenham evoluído rapidamente, ainda carecem de uma série de definições e padrões que sejam amplamente aceitos. Um dos aspectos onde esta deficiência é mais visível é a técnica de modelagem de *workflow*.

O desenvolvimento de sistemas de *workflow* envolve obrigatoriamente um certo número de etapas, dentre elas estão:

- **Modelagem de Processos e Definição de *Workflow***: Requer modelos de *workflow* e metodologias para capturar o processo como uma definição de *workflow*.
- **Reengenharia de Processos:** Requer metodologias para a otimização dos processos. Esta área é basicamente preocupação dos profissionais e pesquisadores de administração organizacional.
- **Implementação e Automação de *Workflow*:** Requer metodologias e tecnologias para utilizar sistemas de informação e participantes humanos para implementar, escalonar, executar e controlar as atividades definidas no *workflow*.

A função de um modelo de *workflow* é capturar um determinado conjunto de elementos relevantes da realidade e expressá-los em um certo formato [AMA 97]. A modelagem de *workflow* consiste em representar, através de algum formalismo, as atividades que compõem os processos, sua seqüência de execução e seus inter-relacionamentos, assim como os agentes responsáveis por sua execução e os recursos envolvidos [JUN 2000]. Sendo que cada processo adapta-se melhor a um ou a outro modelo existente, não havendo desta forma um único modelo amplamente aceito.

A modelagem do *workflow* pode ser baseada em comunicação ou em atividades. As metodologias baseadas em comunicação enxergam o trabalho como um conjunto de interações humanas bem definidas. Já as metodologias baseadas em atividades

enxergam o trabalho como uma seqüência de atividades, onde cada uma recebe um conjunto de entradas e produz um conjunto de saídas [NIC 98].

Como exemplo de modelagem baseada em atividades temos o modelo de Casati/Ceri e o Modelo de Gatilhos.

Com o modelo definido e convenientemente alimentado com as informações necessárias, o sistema de gerência de *workflow*, baseado nestas informações coordena a execução das atividades.

Vejamos agora alguns dos principais modelos existentes.

4.5 Modelagens para Sistemas de Informação de Escritórios

Segundo [EDE 94] os modelos resultantes da especificação de um SIE podem ser classificados conforme visões do próprio escritório ou dos diferentes aspectos do trabalho nele desenvolvido. E podem ser classificados como:

- **Modelos Baseados em Bases de Dados.**
- **Modelos Baseados em Processos.**
- **Modelos Baseados em Agentes.**
- **Modelos de Tomada de Decisão.**
- **Modelos Comportamentais.**

4.6 Modelagens Baseadas em Atividades

As metodologias baseadas em atividades caracterizam-se pela existência de uma atividade e, eventualmente, pré-condições e/ou pós-condições para a sua ocorrência. A principal fundamentação teórica para estes modelos de *workflow* está na Rede de Controle de Informações (*Information Control Net* – ICN), desenvolvida na década de 70 pelo Dr. Clarence Ellis, modelo este que foi idealizado para descrever atividades realizadas em escritórios.

4.6.1 Modelo de Barthelmes/Wainer

As ocorrências dos sistemas de *workflow* são assíncronas, isto é, não é possível antecipar o exato momento de sua ocorrência. Dependendo do momento da ocorrência do evento, respostas diferentes podem ser geradas [SIZ 99]

Este modelo proposto por Barthelmes e Wainer suporta eventos assíncronos tanto no momento da modelagem como no momento da execução do *workflow*, baseado na idéia de se ter transações entre eventos e não entre atividades. Assim, eventos externos e

estados de espera são representados de uma maneira uniforme, mesmo que estes estados não tenham nenhuma atividade associada (são puramente uma saída para o sincronismo do *workflow*).

A principal diferença entre o modelo proposto e os demais modelos baseados em trigger está em se lidar com eventos assíncronos tanto na especificação como na execução do workflow.

Usando-se atividades (modelo de gatilhos), não se pode representar um estado de espera entre as duas atividades. Por outro lado, usando-se eventos pode-se inserir um evento entre as duas atividades, demonstrando uma seqüência de disparos de atividades.

4.6.2 Modelo de Casati/Ceri

O modelo Casati/Ceri é considerado um dos modelos mais completos para a especificação de *workflows*, sendo representado através de uma linguagem gráfica, o que facilita a compreensão e análise, consiste de um conjunto de tarefas e conexões entre elas. Descreve-se no esquema a ordem de execução das tarefas, o encarregado da sua execução e as operações que podem ser executadas. Neste modelo, a especificação de tarefa pode incluir consultas de bancos de dados através de comandos da linguagem SQL.

A definição dos processos é modelada através de uma linguagem de descrição de *workflow* (*Workflow Description languages - WFDL*), que descreve as atividades que são realizadas durante e a sua ordem de execução. A linguagem textual serve para descrever as tarefas e os mecanismos usados na estrutura do fluxo de um esquema de *workflow* pela composição de objetos [NIC 98]. Vejamos algumas características deste modelo:

- Descrição formal do comportamento interno do *workflow*, com a definição, interação e cooperação de atividades,
- Relacionamento entre *workflow* e seu ambiente, com a atribuição de atividades a atores;
- Acesso a bases de dados externas (através de comandos SQL),
- Noções de modularização de tarefas (supertarefas)
- Prevê a representação do tratamento de exceções.

Os elementos presentes no modelo de Casati/Ceri são:

- **Tarefa**: É uma unidade elementar de trabalho cujo conjunto completo constitui a finalidade do *workflow*. É detalhada através das seguintes características:
  - **Nome**: Identifica a tarefa. É o único valor de preenchimento obrigatório;
  - **Descrição**: Objetivo da tarefa, expresso em linguagem natural;
  - **Pré-condições:** Uma expressão lógica que deve ser avaliada antes da realização da tarefa.
  - **Ações:** São comandos em WFDL que definem como os dados serão manipulados, para efetivamente realizar a tarefa.
  - **Exceções:** Lista de valores <Exceção, Reação>, indicando as atitudes a serem tomadas quando ocorrerem eventos anormais ao fluxo comum.

- **Conexões entre Tarefas:** Estipulam os tipos de roteamento possíveis entre duas tarefas a serem executadas. As conexões podem ser do tipo:
  - **Causalidade**: O término de uma tarefa habilita o início de sua sucessora.
  - **Forks**: O término de uma tarefa habilita a execução de diversas sucessoras. Essa relação pode ser de quatro formas:

  Total: O término de uma tarefa habilita a execução de todas as suas sucessoras.
    - **Não-determinístico**: O término de uma tarefa habilita a execução de um número K de sucessoras, escolhidas de modo não-deterministico;
    - **Condicional**: Habilitam-se somente as sucessoras com condição verdadeira.
    - **Condicional com exclusão mútua**: Avalia-se uma condição e somente uma sucessora é habilitada.
  - **Joins**: Nos *joins* uma tarefa é precedida por um conjunto de outras denominadas predecessoras. Podem ter as seguintes formas:
    - **Total**: Uma tarefa (M) é habilitada somente após o término de todas as suas predecessoras;
    - **Parcial**: Uma tarefa (M) é habilitada após o término de k predecessoras;
    - **Iterativo**: Uma mesma tarefa (M) é habilitada a cada término de k predecessoras, o que pode gerar a execução paralela de várias instâncias da mesma tarefa.
  - **Símbolos de Início e Fim:** Os símbolos de início e fim indicam respectivamente o início e o término de uma instância de *workflow*.
  - **Supertarefa:** É o agrupamento de várias tarefas que diminui a complexidade do *workflow*, pois utiliza os conceitos de modularização, proporcionando maior facilidade nos casos de alterações no modelo.
  - **Multitarefa:** É utilizada para definir um conjunto de tarefas que desenvolvem exatamente o mesmo trabalho em paralelo, mas com atores diferentes.
  - **Agentes:** Uma tarefa pode ser completamente automatizada ou ser atribuída a um agente.
  - **Templates:** Para complementar o modelo de Casati/Ceri, em [SIZ 99] foi proposta uma extensão baseada em *templates*, possibilitando a atribuição de responsabilidades, tratamento de exceções e determinação de pré e de pós-condições, possibilitando também que as informações referentes às tarefas sejam mais detalhadas, principalmente no que diz respeito aos agentes do sistema.

4.6.3 Método CIMOSA de Modelagem de Negócios

CIMOSA é a sigla para *Computer-Integrated Manufactuting – Open-Systems Architecture*, sendo um método de modelagem de empresas. É um método muito abrangente, totalmente consolidado, aceito e utilizado em industrias de manufaturas de todo o mundo, seguindo 4 visões:

- De Organização;
- De Recursos;
- De funções;
- De Informações

4.6.4 Modelagem Conceitual de Esquemas de Workflow

Este modelo de representação de *workflow* se propõe criar um modelo conceitual de representação de processos independente de implementação. Com ele, os seus criadores esperam permitir o desenvolvimento da definição de processo livre dos vícios e limitações que podem existir em uma ferramenta comercial de *workflow*. Desta forma, muitas das informações que muitas vezes não podem ser representadas neste ambiente poderiam ser documentadas num modelo conceitual.

O modelo de representação conceitual proposto por Meyer-Wegener e Bohn apresenta uma abordagem que utiliza uma descrição conceitual do modelo de processos, independente de implementação, que não possui os inconvenientes de um modelo vinculado a um sistema de gerência de *workflow*, que muitas vezes não prevê a representação de algumas informações sobre o processo [BRE 2001]. Uma vez modelado o processo de *workflow*, o modelo conceitual poderia então ser mapeado semi-automaticamente para um determinado produto de *workflow*, onde ocorreria a sua implementação. Para isso, o modelo conceitual é dividido em três estruturas:

- **Tipo de Tarefa:**Um tipo de tarefa é representado no modelo por um retângulo com um nome. Todas as tarefas do processo de *workflow* devem aparecer nesta estrutura representada por este retângulo. As tarefas são organizadas hierarquicamente, representando uma decomposição funcional. Na falta de uma definição mais detalhada, convém definir a atividade proposta como sendo uma atividade do tipo “sem implementação”. Através do operador de agregação, o modelo também define as atividades do tipo sub-fluxo, permitindo representar uma atividade como sendo a composição de outras atividades.
- **Configuração de Tipo de *Workflow*:** A estrutura de tipo de tarefa permite que se represente as atividades que compõem um processo. Muitas vezes, porém, uma representação única do modelo do processo pode se apresentar extremamente complexa para a discussão com os usuários do sistema, por exemplo, ou extremamente simples para permitir a implementação do processo. Logo, a representação de diversos tipos de modelo de processo de acordo com as necessidades existentes, torna-se uma possibilidade bastante interessante. Para criar estas diversas representações sobre um processo, deve-se marcar de alguma forma, na estrutura de tipo de tarefa, quais atividades fazem parte de uma certa configuração de *workflow* e quais atividades fazem parte de outra.
- **Estrutura de Execução de Instruções:** A estrutura de execução de instruções define a ordem em que as tarefas são executadas. Os elementos definidos por este modelo podem ser Primitivas de controle de fluxo, Construtores de controle de fluxo ou Instruções de execução. As primitivas de controle de fluxo representam as transições entre as tarefas e normalmente são

representadas como linhas diretas entre elas. Os construtores de controle de fluxo combinam um conjunto de primitivas e normalmente possuem um nome para identifica-las. São exemplos o AND-Split, OR-Split, AND-Join e OR-Join definidos pela WfMC, que representam todos os tipos de distribuição e junção identificados nas dependências entre atividades e que podem ser modelados através destas primitivas. As instruções de execução permitem a representação dos fluxos entre atividades através das instruções pré-definidas.

4.6.5 Modelo de Gatilhos

O modelo de gatilhos é uma técnica simples proposta por Stef Joosten, que visa auxiliar a análise de processos de *workflow*. Neste modelo o comportamento de um sistema de *workflow* é descrito através de gatilhos que geralmente são modelados utilizando-se Redes de Petri.

Um evento no modelo de gatilhos é a pré-condição da atividade (um evento dispara atividades) e também é a pós-condição de atividades (eventos ocorrem como resultado da execução de uma atividade). Este comportamento é chamado de gatilho (*triggering*).

O modelo pode conter até três diferentes tipos de atividades que são representados pelos seguintes símbolos:

- **Retângulo**: Representam atividades que podem ser particionadas.
- **Circulo**: Representam atividades atômicas, que não podem ser particionadas.
- **Triângulo**: Representa um ponto de sincronização, utilizado para sincronizar os gatilhos, ou seja, direcionar os gatilhos para várias atividades ou juntar vários gatilhos para a execução da próxima atividade. Um arco entre atividades significa que a atividade sucessora pode ser disparada por um dos eventos da atividade predecessora. Com o triangulo, é possível representar os conectores de distribuição total e junção total.

Na representação utilizada no modelo de gatilhos cada atividade é representada por um retângulo e contém o nome da atividade. Uma seta direcionada para a atividade significa que a atividade pode ser disparada pelos eventos que ocorreram como resultado do final de outra atividade. Esse modelo é dividido em colunas, cada qual contendo as atividades associadas com um papel particular.

4.6.6 Modelo de *Workflow* do produto “Oracle *WorkFlow*”

A ferramenta de *workflow* da Oracle, “Oracle *Workflow*”, possui um modelo próprio de representação que pode ser criado a partir do programa Oracle *Workflow Builder* [BRE 2001].

O modelo de representação de *workflow* da Oracle é composto pelos seguintes elementos:

- **Notificações**: Representam atividades que são enviadas para os participantes do *workflow*. São sempre executadas por agentes, correspondendo às atividades do tipo sem implementação. As notificações podem ser vinculadas a um usuário do sistema, um papel ou um atributo do *workflow*.
- **Funções**: Correspondem a programas, internos ou externos, que são executados pelo sistema de *workflow*. São responsáveis pela execução das atividades automáticas do *workflow*.
- **Processos**: Representam a definição do processo de *workflow*. Todo o processo deve possuir uma atividade de início e pelo menos uma atividade de fim. Dentro de um processo podem existir notificações, funções e outros processos (que, quando dentro de outro processo, tornam-se sub-processos de *workflow*). Equivalente a atividade do tipo sub-fluxo.
- **Lookup Type:** Permite a representação no sistema de tipos de dados que podem ser utilizados, por exemplo, para criar as rotas entre os processos.

As dependências entre as atividades são representadas por linhas que são traçadas entre quaisquer dois elementos do processo. Sobre cada rota pode-se colocar uma condição que, quando verdadeira, faz com que o processo siga por aquela rota.

Algumas funções especiais, que já vem pronta com a ferramenta, permitem a implementação de conectores de junção e distribuição, disponibilizando para o projetista diversas opções de roteamento de processo.

4.6.7 Redes de Petri

As duas possíveis abordagens para modelagem de sistemas por Redes de Petri são:

- **Abordagem F**: Seu conceito fundamental é o fenômeno (estado ou transição), serve como abordagem conceitual para melhor compreensão da abordagem C/E
- **Abordagem C/E**: Seu elemento fundamental é a condição e o evento, destina-se à aplicação prática.

Fenômenos do tipo estado são classificados em classes que recebem a denominação de condição e os fenômenos do tipo transição são classificados em classes que recebem a denominação de evento.

A utilização de Redes de Petri para se modelar *workflows* favorece a formalização de importantes conceitos como: tarefas, procedimentos, recursos, atividades de controle, *jobs* e outros, trazendo semântica ao *workflow* e ao WFMS, deixando-os mais claros e compreensíveis.

Esta semântica formal e a modelagem explicita dos estados destacam-se dentre as vantagens da modelagem de sistemas de *workflow* com Redes de Petri. Através da semântica formal pode-se passar o modelo obtido diretamente e sem ambigüidades para um WFMS que compreenda *Petri-Nets*. Já o estado de uma determinada instância do *workflow* pode ser modelado explicitamente em uma Rede de Petri, permitindo a modelagem de diversas situações onde a habilitação e a execução representam momentos diferentes.

4.6.8 Modelo TF-ORM

O modelo TF-ORM (*Temporal Functionality in Objects with Roles Model*) [EDE 94] é um modelo conceitual, orientado a objetos que utiliza o conceito de papéis para representar os diferentes comportamentos dos objetos. Adicionalmente introduz o suporte a propriedades com variação dinâmica de valor e definidas sobre domínios temporais, bem como a regras de transição de estado e restrição de integridade.

Este modelo é uma extensão do modelo de dados F-ORM, o qual visa especificamente a representação de sistemas de informação de escritórios, através da extensão do modelo de dados ORM.

Através do modelo TF-ORM, o desenvolvedor pode construir o esquema de dados de um sistema de informação utilizando classes para encapsular aspectos estáticos e dinâmicos da aplicação, modelar os aspectos tempo-dependentes utilizando domínios temporais e linguagem de lógica temporal, regras de transição de estado e restrições de integridade.

Este modelo possui uma linguagem de consulta baseada na linguagem SQL, com alguma influência do Tquel e apresenta os seguintes elementos:

- **Classes e Papéis:** Uma classe TF-ORM é definida por um nome c_n, que a identifica em toda a hierarquia de classes da aplicação, e por um conjunto de papéis R_i, representando os diferentes comportamentos dos objetos desta classe. São identificados três tipos de classes: classes de recursos, classes de agentes e classes de processos. Cada papel ***R*** é representado por um nome Rn_i, que o identifica univocamente dentro da classe a que pertence, e contém:
  - Um conjunto de propriedades P_i, implementadas como atributos;
  - Um conjunto de estados S_i, que este objetos pode assumir quando desempenhando este papel;
  - Um conjunto de mensagens M_i, que as instâncias deste papel podem enviar ou receber;
  - Um conjunto de regras de transição de estado e regras de integridade Ru_i, descrevendo o comportamento dinâmico das instâncias do papel, através de restrições aos valores possíveis das propriedades.

  Nos papéis definidos em classes de agentes as decisões D_i que podem ser tomadas pelas instâncias deste papel. As decisões aparecem nas regras de transição de estado e têm por finalidade condicioná-las a uma tomada de decisão pelo agente. Toda classe TF-ORM inclui a definição de um papel especial R_0, chamado papel básico (*base role*). Neste papel, são descritas as características iniciais de toda instância desta classe e as suas propriedades globais. Este papel apresenta somente propriedades (herdadas por todas as instâncias dos demais papéis) e regras (que controlam as instâncias dos outros papéis). Seus estados são pré-definidos (*active e suspended*), assim como as mensagens que podem ser utilizadas em suas regras (mensagens de criação, suspensão, retomada e término de instância de uma classe e de um papel).
- **Instâncias**: No modelo TF-ORM existem dois tipos distintos de instâncias: instâncias de classes, também denominadas objetos e instâncias de papéis.
- **Identidade:** No modelo TF-ORM, da mesma maneira que no modelo de objetos com papéis, é definido um identificador de objeto (*oId*) e um identificador de instância de papel (*rId*). Desta forma, um objeto fica perfeitamente caracterizado pelo seu *oId*, ao passo que uma instância de papel, é caracterizada pelo par *oId.rId*.

- **Propriedades:** O modelo TF-ORM identifica dois tipos básicos de propriedades, conforme a possibilidade de apresentarem variação com o tempo: As propriedades estáticas e as propriedades dinâmicas.
- **Estados:** Durante sua existência, uma instância de papel pode assumir diversos estados. Um estado de uma determinada instância de papel indica o atual contexto de execução do objeto naquele papel. Os nomes dos estados devem ser únicos dentro de um determinado papel, sendo que em papéis diferentes é possível a ocorrência de estados com o mesmo nome. Como exemplo de nomes de estados podemos citar: *Trabalhando, férias, licença, aposentado* que podem ser utilizados como estados de um funcionário. Existem também estados pré-definidos que atuam no papel básico controlando a execução dos objetos e instâncias de papeis. São eles: *Active* (quando o objeto está em execução) e *suspended* (quando o objeto ou instância de papel está impedida de enviar e receber mensagens).
- **Mensagens:** Todas as mensagens relevantes para a descrição do comportamento da instância, através da ativação de mudanças de estado, devem ser consideradas no TF-ORM. Essas mensagens devem ser listadas, com seus parâmetros e origem ou destino. Toda mensagem possui uma direção, ou seja, pode ser recebida ou enviada pela instância. Dessa forma, utiliza-se o termo *from* (mensagem de entrada) e *to* (mensagem de saída) para especificar a direção. As mensagens responsáveis pelo controle da criação e remoção de objetos e instâncias de papeis, bem como pelo controle de mensagens e regras, estão pré-definidas no papel básico das classes:
  Create_object (<classe>,<oId>): cria uma instância (objeto) de classe, retornando seu oId;
  - ***Suspend_object* (<*oId*>) e *resume_object* (<*oId*>):** suspende e prossegue a execução do objeto identificado por *oId*, transacionando-o entre os estados *active* e *suspended*;
  - ***Kill*(<*oId*>):** termina a execução do objeto identificado por *oId*;
  - ***Add_ role* (<*oId*>, <*papel*>, <*rId*>):** cria uma instância de papel para o objeto identificado por *oId*, retornando seu *rId*;
  - ***Suspend_role* (<*oId*>,<*rId*>) e *resume_role* (<*oId*>,<*rId*>):** suspende e prossegue a execução da instância de papel *rId* do objeto *oId*, transacionando-a entre os estados *active* e *suspended*;
  - ***Terminate_role* (<*oId*>,<*rId*>):** termina a execução da instância *rId* do objeto *oId*;
  - ***Forbid_role* (<*oId*>,<*papel*>) e *allow_role* (<*oId*>,<*papel*>):** proíbe e permite novamente a instanciação de papel no objeto *oId*;
- **Regras de Transição de Estado:** As regras de transição de estado definem o comportamento ativo dos objetos e instâncias de papel, modelando as transições de estado para estes.
- **Regras de Integridade:** As regras de integridade devem sempre ser satisfeitas por todas as instâncias do papel.
- **Representação do *workflow* através do TF-ORM:** Em [NIC 98] é proposta a utilização do TF-ORM para modelagem de *workflows*. A correspondência entre os conceitos de um *workflow* e a sua representação no TF-ORM pode ser apresentada conforme mostrado nas tabelas abaixo.

TABELA 4.1 Classes de Processos de um *Workflow*

Workflow	**TF-ORM**
Processo	Classe de processo
Atividade	Papel de uma classe de processo
Agente responsável por uma atividade	Atributos na criação da instância de papel
Propriedades comuns a todas as atividades	Propriedades do papel básico da classe
Propriedades específicas de uma atividade	Propriedades do papel
Operações executadas em uma atividade	Mensagens e decisões do papel
Estados inicial e final de uma operação	Estados do papel
Sincronismo entre as atividades a serem executadas	Regras de transição de estados
Condições gerais de integridade	Regras de integridade

TABELA 4.2 Classes de Agentes de um *Workflow*

Workflow	**TF-ORM**
Pessoa	Classe de agente
Pessoa executando uma atividade	Papel de classe de agente
Propriedades comuns a todos os comportamentos do agente	Propriedades do papel básico da classe
Propriedades especificas de um determinado comportamento	Propriedades do papel correspondente
Decisões que o agente pode tomar	Decisões do papel correspondente
Ações que o agente pode executar	Mensagens do papel
Estados inicial e final de uma operação	Estados do papel
Evolução do agente	Regras de transição de estados

TABELA 4.3 Classes de Recursos de um *Workflow*

Workflow	**TF-ORM**
Recurso manipulado em um processo	Classe de recurso
Diferentes aspectos do recurso	Papel da classe de recurso
Propriedades comuns a todos os comportamentos dos recursos	Propriedades do papel básico da classe
Propriedades especificas de um determinado comportamento	Propriedades do papel correspondente
Ações executadas sobre o recurso	Mensagens do papel
Estados inicial e final de uma operação	Estados do papel
Evolução do recurso	Regras de transição

- **Sincronização de Tarefas:** O TF-ORM realiza o sincronismo entre as atividades por meio das regras de transição de estados utilizando mensagens que podem representar as seguintes evoluções:
  - **Seqüencial**: As atividades são disparadas obedecendo a uma ordem de execução;
  - **Paralela**: Durante a sua execução, uma atividade pode disparar a execução de outra atividade (do mesmo ou de outro processo);
  - **Convergente(join)**: Uma atividade pode ser ativada em função de diversas mensagens recebidas;
  - **Divergente(Fork)**: Diferentes atividades podem ser ativadas simultaneamente por uma mesma atividade;

- **Condicional**: Quando uma atividade é ativada através de mensagem oriunda de uma condição associada a uma regra de transição.

- Extensões do TF-ORM: Algumas extensões foram propostas em [NIC 98] que permitem ao TF-ORM representar de uma forma mais abrangente os aspectos relacionados a um *workflow*, são elas:
  - **Extensões na Classe de Agentes:** As extensões propostas para a classe de agentes têm o objetivo de especificar quais as atividades e decisões que são de responsabilidade ou são executadas pelo agente em uma ou mais atividades. A sintaxe *executor_tasks* permite representar todos os processo (*ProcessClass*) e papéis (*Role*) nos quais o agente é executor. A sintaxe *responsible_tasks*, permite representar todos os processos (*ProcessClass*) e papéis (*Role*) nos quais o agente é responsável. A sintaxe *decision_task* representa todas as decisões tomadas pelo agente no processo ($ProcessClass_{m/n}$) mais especificamente em um determinado papel ($Role_i$) deste processo.
  - **Extensões para a Classe de Processo:** As extensões realizadas para a classe de processo foram:

Definição de responsabilidades: Qual o agente executor é o responsável para uma tarefa ao criar a instância de um papel;
Listagem das decisões: Permiti que decisões tomadas pelo agente sejam representadas na classe de processos, monitorando e determinando as interações dos agentes sobre os processos.

- **Definição do número de mensagens**: Foi proposta também uma extensão relacionada ao sincronismo. Permite restringir o número de mensagens de chegada, necessárias para que ocorra uma transição. Esta regra representa a situação em que diversas mensagens (n) podem ser recebidas, quando um determinado número (k < n) destas mensagens são necessárias para que ocorra a transição. Para este caso a chegada de mensagens pode ser aleatória.

4.6.9 Modelo Wide

O projeto WIDE é um projeto de pesquisa e desenvolvimento que busca estender a tecnologia de banco de dados ativos e distribuídos, de forma a melhorar os sistemas orientados a aplicação através de técnicas de *workflow* [BRE 2001]. O projeto é desenvolvido por um consórcio que possui parceiros de diferentes nacionalidades, tais como Espanha, Itália e Holanda.

O modelo WIDE é composto de três diferentes modelos: O modelo Organizacional; O Modelo de Informações; O Modelo de Processos.

O Modelo de Organização busca descrever os setores da organização que estão envolvidos no *workflow*. O modelo descreve, através de diagramas, como são distribuídas as funções e responsabilidades dentro da empresa, visando auxiliar o modelo de processos na alocação de atividades entre os seus participantes. A sua estrutura encontra-se fortemente ligada a estrutura de papéis, criada para dar suporte aos processos de *workflow*. Esta separação divide as entidades previstas no modelo do processo das entidades previstas no modelo da organização.

O Modelo de Informação O modelo de informação identifica e descreve os elementos de documentação envolvidos na definição do processo de *workflow*. Da mesma forma que o modelo de organização, este modelo procura ser o mais independente possível. Ele é composto por:

- **Variáveis de Informação**: São elementos básicos de informação que estão disponíveis em todas as tarefas do modelo de processo.
- **Variáveis de Informação Compartilhadas**: São variáveis externas ao modelo de processo, muitas vezes independentes de uma aplicação particular de *workflow*.
- **Informações Trocadas Através de Elementos de Documentação**: Um elemento de documentação é um conjunto de informações utilizadas, criadas ou modificadas por um usuário para completar uma tarefa.

Já o Modelo de Processo descreve como os processos de *workflow* evoluem do seu estado inicial até o seu estado final. Segundo o modelo WIDE, um processo pode ser composto por:

- **Tarefas**: São as unidades elementares de trabalho, que juntas formam o *workflow*. O motor do *workflow* é o responsável por identificar quais tarefas devem iniciar num determinado momento e atribuir um agente para cada uma das tarefas disparadas de acordo com as políticas estabelecidas na definição do processo.
- **Conectores**: Descrevem a interação entre as tarefas. Pela notação do modelo, uma tarefa A só pode ter uma conexão de entrada e uma de saída. Quando o fluxo não segue estas regras, utilizam-se conectores, que podem ser dos tipos Fork ou Junção.
- **Símbolos de Início e Fim**: Marcam o início e o fim de um processo. Cada processo deve ter somente um símbolo de inicio, mas pode ter vários símbolos de fim.
- **Conector de Espera (Wait):** Faz com que o processo pare no ponto em que está e fique aguardando o acontecimento de um evento externo que faz com que o processo continue a sua execução.
- **MultiTarefa:** A multitarefa representa uma tarefa que é distribuída entre diversos participantes que devem realizar a mesma tarefa em paralelo. Em tempo de modelagem, a multitarefa é vista como uma tarefa única, já na sua execução ela é enviada para diversos atores em paralelo.
- **Sub-processo:** Permite a modularização do trabalho, através da divisão do processo em partes menores, que podem conter todos os componentes acima, alem de outros sub-processos.

4.6.10 Modelo de Referência da WfMC

O modelo de referência de *workflow* foi desenvolvido pela *Workflow Management Coalition* (WfMC) a partir da estrutura de aplicação de *workflow* genérica.

Neste modelo são definidas cinco interfaces entre componentes, além de uma interface sobre o serviço de execução de *workflow*, denominada WAPI (*Workflow API and Interchange Formats*). Esta interface incorpora especificações para capacitar

interoperabilidade entre os diferentes componentes do WFMS e as aplicações [WOR 2002]. As interfaces definidas são:

1. Definição de Processo: Procedimentos que são seguidos na implementação do *workflow*, e os recursos (pessoas, sistemas, grupos) que executam o trabalho. Algumas vezes o *workflow* é definido separadamente em ferramentas de modelagem e simulação, o *workflow* é carregado na máquina através da interface 1. Isto também permite que o *workflow* seja movido de um sistema para outro.

2. Interface Cliente: A forma como o programa de aplicação invoca o *workflow*, solicitando a próxima tarefa ou completa o trabalho atual.

3. Aplicações Invocadas: Programas que podem ser invocados pelo sistema de *workflow*. A interface de programação da aplicação de *workflow*, WAPI, combina a interface 2, onde as aplicações invocam o *workflow*, com a interface 3, onde o *workflow* invoca aplicações. Isto inclui interfaces para sistemas de dados como sistemas de imagens, controle de documentos e correio.

4. Serviços de Workflow Externos: Relaciona-se a interoperabilidade entre sistemas de *workflow* independentes. A interface 4 da WfMC inicialmente usada como MIME, define a interface para uso de ferramentas de e-mail e tecnologias. Como a tecnologia XML é emergente foi criado o Wf/XML utilizando a tecnologia XML. A intenção é que esta interface possa ser utilizada para:

a. **Processos macroscópicos**, tais como solicitação de um produto ou serviço de outra organização, possibilitando controlar completamente o processo.
b. **Processos microscópicos**, tais como escolher um item individual em um pedido, por um sistema manual externo ou por uma interface automatizada do sistema de estoque.

5. Administração e Monitoramento do histórico: de cada caso e de todo o processo executado.

a. Casos individuais devem incluir a pessoa ou recurso que foi designado para a tarefa, com a data, hora e disposição. Isto será utilizado para relatórios e auditorias, mas pode também ser utilizado para controle do processo, tais como situação onde duas aprovações por diferentes pessoas foram solicitadas. (O processo precisaria checar o histórico para ter certeza que a solicitação da segunda aprovação não foi feita pela mesma pessoa que deu a primeira aprovação)
b. Carga de trabalho global inclui um “instantâneo on line” da quantidade de trabalho esperando para ser processada e o status do sistema, bem como o histórico que será utilizado para obter um relatório da produtividade global.

Além destas cinco interfaces, foi definida uma interface sobre o serviço de execução de *workflow*, denominada WAPI (*Workflow API and Interchange Formats*). Esta interface incorpora especificações para capacitar interoperabilidade entre os diferentes componentes do WFMS e as aplicações [WOR 2002]. Sendo assim, os serviços de *workflow* podem ser implementados de diferentes formas, desde que sejam oferecidas interfaces que traduzam os métodos internos de cada produto de *workflow* para os métodos padronizados pela WfMC. Essa interoperabilidade presente no modelo é obtida através de um *metamodelo* que contém as informações que devem ser

armazenadas sobre o *workflow*, uma linguagem padrão para interoperabilidade chamada WPDL (*Workflow Process Definition Language*) [SIZ 99] e uma API, WAPI, que provê a manipulação dos atributos das entidades de definição de processos.

Abaixo temos o diagrama do modelo de referencia de **workflow** da WfMC.

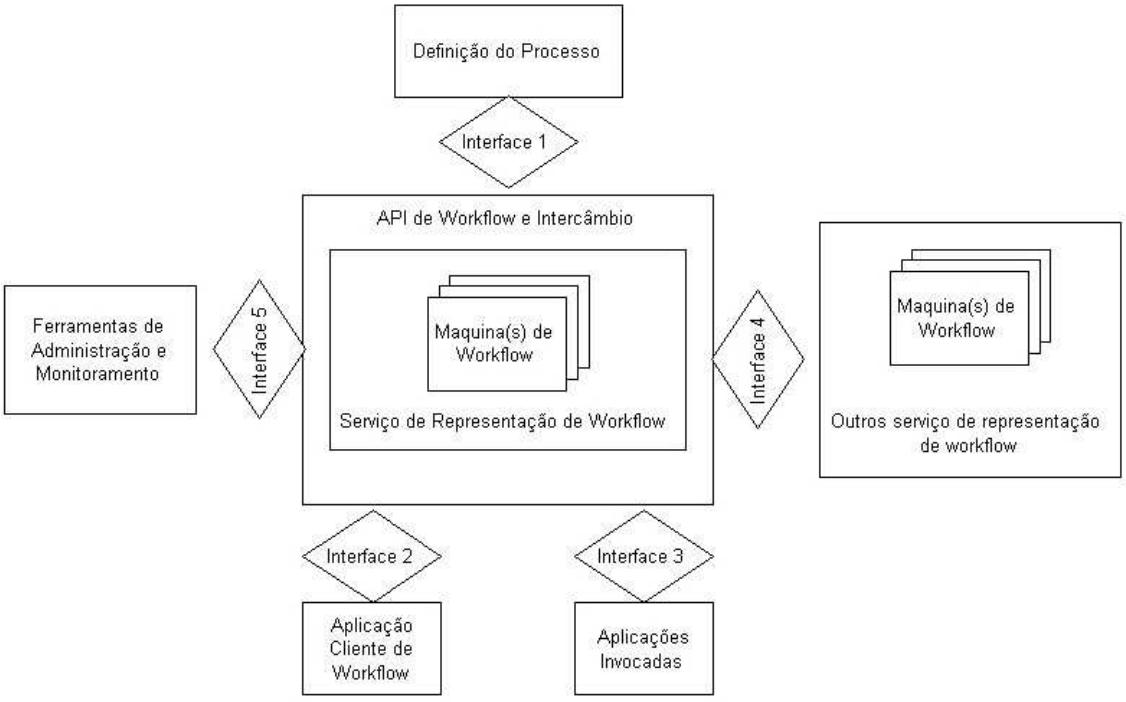


FIGURA 4.2 Diagrama do Modelo de *Workflow* da WfMC

Sendo assim, os serviços de *workflow* podem ser implementados de diferentes formas, desde que sejam oferecidas interfaces que traduzam os métodos internos de cada produto de *workflow* para os métodos padronizados pela WfMC. Essa interoperabilidade presente no modelo é obtida através de um *metamodelo* que contém as informações que devem ser armazenadas sobre o *workflow*, uma linguagem padrão para interoperabilidade chamada WPDL (*Workflow Process Definition Language*) [SIZ 99] e uma API, WAPI, que provê a manipulação dos atributos das entidades de definição de processos.

O modelo da WfMC é o modelo ideal para ser estendido, por sua crescente aceitação, por parte dos fabricantes de software, como sendo um padrão de interoperabilidade, os processos representados através dele seriam reconhecidos pelas ferramentas existentes no mercado que aceitam este padrão.

Para manter a interoperabilidade a extensão proposta neste trabalho visa não alterar o modelo, aproveitando-se da opção de utilizar atributos extendidos onde for possível, de acordo com o modelo da WfMC e as demais informações necessárias a obtenção da meta deste trabalho, ficarão como informações da máquina de workflow em tabelas que apesar de estarem no mesmo banco, não fazem parte do modelo.

No modelo da WfMC existe uma entidade de definição de processo de *workflow* , que define os elementos que fazem parte do *workflow*. Contém as definições ou declarações, respectivamente, de atividades, transições, aplicações e de dados relevantes do processo. Os atributos desta entidade armazenam dados sobre a sua identificação, criação, simulação, responsabilidade e direito de acesso.

A definição de Atividades do Processo de Workflow, é utilizada para definir cada uma das atividades elementares que fazem parte do processo de *workflow*. Esta entidade

possui atributos que armazenam as informações sobre o controle da atividade, as suas alternativas de implementação, a forma como será identificado o participante que realizará a atividade, os dados utilizados durante o processamento da atividade (como a sua prioridade, por exemplo) e os dados utilizados para a simulação de processos. Uma atividade pode ter a função de um conector, chamadas de atividades de roteamento, ou assumir outros quatro tipos possíveis de implementação: Aplicação, Sub-fluxo, Loop ou Sem Implementação. Através das atividades de roteamento são implementados alguns tipos de conexões entre atividades, como:

- **AND JOIN**: Equivalente a junção total, esta atividade aguarda que todas as atividades anteriores tenham se encerrado para iniciar a próxima.
- **XOR JOIN**: Equivalente a junção parcial, esta atividade necessita que somente uma das atividades anteriores seja encerrada para continuar o fluxo.
- **AND SPLIT**: Equivalente aos dois tipos de distribuição, total e condicional. Permite a ativação de várias atividades sucessoras após o término da predescessora. A transição para cada uma das atividades sucessoras pode ou não estar vinculada a uma condição. Caso esteja, a condição deve ser verdadeira para que a sua atividade seja disparada.
- **XOR SPLIT**: Possibilita a definição de uma condição para cada atividade sucessora. A diferença do XOR SPLIT para o AND SPLIT é que nele, somente uma atividade sucessora pode ser disparada.

Uma atividade da WfMC possui atrelada a si atributos que auxiliam o controle da execução da atividade, como:

- **Limite**: Permite especificar o tempo máximo que a atividade possui para ser executadas;
- **Tempo de Espera**: Armazena a média de tempo que a atividade permanece aguardando na lista de trabalho de agente para ser iniciada:
- **Tempo de Trabalho**: Armazena a média de tempo que o agente realmente gasta para realizar a atividade:
- **Duração**: Armazena o tempo médio que atividade demora a ser finalizada. Corresponde a soma dos tempos de trabalho e de espera. A diferença entre o limite e a duração é que o limite fornece um tempo máximo, enquanto que a duração fornece o tempo médio.

O modelo inclui ainda as Informações de Transição, que descrevem as possíveis transições que podem existir entre as atividades, bem como as condições necessárias para que as transições ocorram. São armazenadas nesta entidade informações sobre a atividade origem e destino da transição e o tipo de transição. Através das ligações entre atividades comuns e atividades de roteamento, é possível representar os diversos tipos de dependência entre atividades.

Os participantes do processo estão declarados na Declaração de Participantes do *Workflow*, a qual provê a descrição dos recursos que podem atuar como participantes das várias atividades na Definição de Processo. Um recurso em particular, o qual pode ser atribuído para executar uma atividade especifica, é especificado como um atributo da atividade, "*participant assignment*", o qual liga a atividade a um conjunto de recursos. A declaração de participante do *workflow* não se refere necessariamente a uma pessoa, mas também pode se referir a um conjunto de pessoas de capacidades ou responsabilidades apropriadas, ou ainda a uma máquina ou qualquer outro recurso não humano que seja mais capacitado para a execução da atividade. O meta-modelo inclui quatro tipos simples de recursos que podem ser definidos com a declaração de participantes do *workflow*. Estas informações são utilizadas para determinar quem são

os responsáveis por executar cada atividade. Segundo a WfMC, uma atividade pode estar atrelada a um(a)

- **Unidade Organizacional**: Representa o gerente ou os membros de uma unidade da empresa.
- **Humano**: Representa um determinado participante do *workflow*. Permite a implementação da alocação pré-determinada.
- **Função**: Permite que o sistema identifique, no momento da ativação da atividade, através da execução de uma função, quem será o agente responsável pela atividade. Corresponde a alocação automática por expressão de busca.
- **Papel**: Representa uma determinada habilidade ou responsabilidade que um ou mais participantes da organização possuem.
- **Sistema**: Representa uma atividade automática, ou seja, uma atividade que é executada por um programa.

A alocação por papel e por unidade organizacional funciona de forma bastante semelhante. A única diferença entre elas diz respeito a forma como os participantes são agrupados. O papel representa as funções que cada funcionário possui, enquanto que a unidade organizacional representa o setor onde ele está alocado dentro da empresa. Em ambos os casos, a escolha do agente responsável pode ser feita através de:

- **Alocação automática**: Que identifica através de algoritmos de balanceamento de carga de trabalho, qual dos agentes habilitado a executar a atividade será o responsável;
- **Disponibilização da atividade para todos os agentes habilitados**: Alocação manual passiva
- **Envio da atividade**: Para um usuário com função gerêncial que escolhera quem será o responsável pela atividade (alocação manual ativa)

Temos ainda a Declaração de Aplicações do *Workflow* que descreve as aplicações de TI que podem ser invocadas pelo serviço de *workflow* para dar suporte ao processamento associado com cada atividade. Tais aplicações podem ser ferramentas industriais genéricas ou qualquer outra disponível no ambiente.

Assim o modelo define os Dados Relevantes para o Workflow que representam as variáveis do processo de *workflow* ou do modelo de definição de *workflow*. Estes atributos normalmente são utilizados para armazenar informações, dados sobre decisões tomadas ou parâmetros que precisam ser passadas entre as diversas atividades do processo.

4.7 Modelagens Baseadas em Comunicação

Esta modelagem enxerga o trabalho como um conjunto de interações humanas bem definidas, representando compromissos realizados entre as pessoas envolvidas. Esta modelagem advém dos estudos de Terry Winograd e Fernando Flores sobre a teoria dos atos da fala. A principal premissa da teoria dos atos da fala é que os seres humanos são fundamentalmente seres lingüísticos: ações por eles executadas aparecem na linguagem, em um mundo construído através da linguagem. Em particular, os seres humanos produzem, na linguagem, conceitos comuns que a fim de tomarem ações em conjunto. A linguagem, por essa visão, não é um sistema para representar o mundo ou

para transmitir pensamentos e informação, mas sim uma ontologia: Um conjunto de conceitos que permite aos seres humanos viver e agir em conjunto [AMA 97].

4.7.1 Modelo de Ações (Action Workflow)

Este modelo assume que o objetivo de um processo é aumentar a satisfação do cliente. Existe um *loop* que representa fielmente a estrutura da comunicação humana, sendo genérico para qualquer situação de trabalho que se deseje modelar e universal, no sentido em que é independente de qualquer cultura, linguagem ou meio de comunicação utilizado para conduzi-lo. Sendo assim, todas as ações realizadas no *workflow* são reduzidas a quatro fases baseadas na comunicação entre o cliente (aquele que solicita que algo seja realizado) e o provedor (aquele que executará algo para cliente) do processo (figura 4.1):

1. **Requisição**: O cliente requisita ao provedor que uma ação seja executada ou o provedor se oferece para executar alguma ação;
2. **Negociação**: O cliente e o provedor concordam sobre a ação a ser executada e definem as condições para a satisfação do cliente (por exemplo, prazo de entrega, nível de qualidade e preço);
3. **Execução**: A ação é realizada (pelo provedor) de acordo com os termos estabelecidos. Ao final desta fase, o provedor declara ao cliente que a tarefa está pronta;
4. **Aceitação**: O cliente relata sua satisfação (ou insatisfação) com a ação realizada. Caso haja insatisfação, até o final do *loop*, a situação deve ser resolvida.

Cada laço de *workflow* entre um cliente e um provedor pode ser acoplado a outros laços de workflow para completar um processo de negócios, sendo que um provedor em um determinado laço de *workflow* pode ser um cliente em outro laço. O modelo resultante revela a rede social na qual um grupo de pessoas, preenchendo diversos papéis, executa um processo.

Especificações de *workflow* que utilizam esta metodologia não indicam quais atividades podem ser realizadas em paralelo ou se há qualquer ação condicional ou alternativa. Tendo em vista que o objetivo do processo de negócio é a satisfação do cliente e a ênfase é dada para tal fato, pode ocorrer que para alguns processos de negócios essa ênfase seja considerada muito superficial e inadequada para o objetivo que se quer alcançar.

FIGURA 4.3 Estrutura do laço de *workflow* do modelo *Action Workflow*

4.8 Considerações Finais

Neste capítulo foi apresentada uma visão geral do estado da arte em termos de modelos de *workflow*. Podemos perceber que foram criados muitos modelos, um para cada tipo de necessidade. Estes modelos estavam interessados em modelar os processos na sua representação, automatização e gerênciamento. Hoje a necessidade vai mais alem agora que esta etapa foi cumprida, a próxima etapa está na padronização de um modelo, o qual atenda as diferentes necessidades de diferentes organizações e que ainda tenha suporte a percepção e cooperação na execução das atividades.

Dentro destes objetivos estão a WfMC com o seu modelo de *workflow* e o projeto CEMT, objetivando o uso desta tecnologia para apoiar o ensino. Por isso o modelo que será utilizado neste trabalho é o modelo da WfMC, adicionando os componentes necessários para a obtenção do suporte a percepção e a cooperação.

5 Proposta

A proposta deste trabalho, um "Modelo de Percepção de *WorkFlow* de Execução", tem como objetivo o monitoramento das atividades durante a execução do *workflow*. O objetivo principal deste trabalho é a obtenção de um modelo de *workflow* com capacidade de expressar e executar conceitos de percepção, monitorando e interagindo com o autor e com os agentes do processo modelado, compatível com o modelo de *workflow* da WFMC "*Workflow Management Coalition*" o que garante a integração com a tecnologia de *workflow* usada no projeto CEMT e contribuirá para a disponibilização de cursos multimídia na Web.

Atualmente não há um trabalho de desenvolvimento que contemple os conceitos de percepção e colaboração com base no modelo da WfMC, este trabalho tem por objetivo o estudo da aplicação de técnicas de monitoramento da execução do *workflow* para a identificação do evento, com compartilhamento das informações e regras de integridade como forma de obter a execução de tarefas em grupo, aplicadas ao modelo da WfMC.

Propondo uma possível solução no aspecto de monitoramento da execução do *workflow*, verificando e registrando cada passo do usuário no sentido de fornecer estatísticas e relatórios de utilização, auxiliando os seus usuários na obtenção de uma maior performance dos objetivos propostos.

O "Modelo de *WorkFlow* com capacidade de percepção de eventos", deve ser capaz de monitorar as atividades durante a execução do *workflow*, expressar conceitos de percepção, monitorar e interagir com o autor e com os agentes do processo modelado, ser compatível com o modelo de *workflow* da WFMC "*Workflow Management Coalition*" o que garante a integração com a tecnologia de *workflow* usada no projeto CEMT.

Isto possibilitará ao Sistema de Gerência de *Workflow* verificar e registrar cada passo do usuário no sentido de fornecer estatísticas e relatórios de utilização, auxiliando os seus usuários na obtenção de uma maior performance dos objetivos propostos.

5.1 Características

Conforme mencionado anteriormente o modelo deve comportar as informações necessárias para representação e interação com os eventos que ocorrem ao longo da execução do *workflow*, permitindo a administração da execução e a obtenção das informações que revelem o "Estado" dos trabalhos e do cumprimento das tarefas dos envolvidos no processo. O modelo deve possuir uma especificação clara e que possa registrar os parâmetros necessários ao cumprimento dos objetivos propostos, bem como uma especificação gráfica que possa representar da mesma forma todos os aspectos envolvidos no sistema. Algumas das características que o modelo em conjunto com a máquina de *workflow* devem apresentar são:

1. **Atividades Atômicas ou Não Atômicas:** Contemplando a possibilidade de agrupar quando necessário 2 ou mais atividades com o objetivos de

obter-se um bloco de atividades afins, que devem ser executadas por um ou mais participantes com ou sem cooperação.

2. **Atividades Programadas:** A serem executadas no futuro.
3. **Estados dos Processos e Atividades**:
    a. Não Iniciado
    b. Rodando
    c. Suspenso
    d. Completado
    e. Abortado
    f. Terminado
4. **Permitir Cooperação**: Entre os participantes na execução das tarefas.
5. **Regras de Cooperação:** Suporte a cooperação através de regras de integridade.
6. **Sistemas de Mensagens:** Entre os participantes de forma on-line e off-line, proporcionando diversos níveis de interação e cooperação.
7. **Suporte à Decisão do Grupo:** Propiciar um ambiente onde uma decisão que deve ser tomada por um grupo de participantes possa ser votada e decidida.
8. **Conferências:** Sessões fechadas de conversa on-line por chat entre determinados participantes.
9. **Sistemas de Coordenação:** Propiciar suporte a coordenação por um ou mais participantes, ou administrador do sistema, de forma que possa gerênciar o processo adequadamente.
10. **Agenda de Tarefas**: Pela leitura do *workflow*, ou pela inserção de tarefas pelos participantes o sistema deverá oferecer uma agenda das tarefas a serem realizadas.
11. **Histórico**: Proporcionar um histórico de todos os eventos ocorridos no sistema em um determinado período.
12. **Tratamento de programação de Tarefas Futuras**: Através da agenda o sistema deve ser capaz de tratar a programação das tarefas futuras.
13. **Controle de Prazos**: Uma forma de monitoramento dos prazos de entrega dos trabalhos, que ao estarem em atraso deve executar uma ação pré-determinada.
14. **Propiciar o Awareness:** Informar Cada participante quanto ao que está fazendo, o que os outros estão fazendo, o que deve ser feito, quem está fazendo o que ele faz, quem pode colaborar com ele, como está o trabalho de quem está colaborando com ele, e troca de informações com qualquer membro do grupo e principalmente com quem está colaborando com ele.
    a. Informar quais são as atividades que devem ser realizadas para que a atividade ou o projeto seja concluído, quais são os prazos, etc.
    b. Através da percepção responder :
        i. Quem realizou dada tarefa? Quando?
        ii. Quem está trabalhando agora? Em que está trabalhando?
        iii. Quem é o responsável por uma tarefa? E o que ainda falta ser feito?
        iv. Relatar as atividades passadas, mostrar o status atual e opções futuras.
    c. Uma mensagem de aviso quando estiver próximo do término do prazo, não apenas quando estiver atrasado. Um atributo de quanto

tempo antes deve ser avisado seria um atributo de configuração do sistema!

d. Elementos de percepção de AWARENESS:
    i. Presença: Quem está no espaço de trabalho?
    ii. Localização: Onde eles estão trabalhando?
    iii. Nível de Atividade: Quão ativo estão os participantes?
    iv. Ações: O que os participantes estão fazendo, e o que devem fazer.
    v. Intenções: Qual é a próxima tarefa a ser feita pelos participantes e onde eles estarão ou executarão.
    vi. Alterações: O Que foi alterado e onde.
    vii. Objetos: Quais os objetos que estão sendo utilizados
    viii. Extensão: O que os participantes podem ver, até onde eles podem chegar.
    ix. Habilidades: O que cada participante é capaz de fazer
    x. Influencia: Onde cada participante pode fazer alterações.
    xi. Expectativa: O que eu posso fazer depois?

15. **Fornecer**:
    a. Tempos de duração, pausa, continuidade.
    b. Número de intervalos em uma atividade, Número de repetições da atividade, estatísticas de uso e ou dificuldade da atividade e ou processo.
    c. Controles de Pendências e Atividades Concluídas, com comunicação para permitir otimização do processo e impedir atrasos.
    d. Controle do Status da atividade, do Participante, do Processo.
    e. Comunicação entre participantes, entre atividades (colaboração) e entre o sistema de controle e as atividades e participantes.
    f. Regras de Integridade e Regras de Obrigatoriedade ou Opcionalidade na seqüência de execução do *Workflow*.
    g. Apresentar histórico de atividades, de ações, de mensagens, de avisos, de versões...
    h. Opções como lembretes, com início em tal data, com tal periodicidade, ate tal data, etc.
16. **Controle de Concorrência**: É necessário para evitar eventuais colisões de ações num espaço compartilhado de trabalho. Este mecanismo geralmente está baseado na resolução de colisões ou na prevenção de concorrência. Num editor cooperativo, uma colisão ocorre quando dois ou mais usuários querem editar uma mesma seção de um documento ao mesmo tempo. Esta situação pode ser impedida através da proibição da mesma edição por dois ou mais usuários ou através da implementação de um mecanismo de ordenação por tempo, definição de papeis, definição de prioridades etc.
17. **Controle de versão**: Quando um usuário estiver editando um documento, é importante que os outros usuários estejam cientes de cada nova versão gerada. O controle de versão é importante para garantir a integridade de cada versão manipulada. Portanto, uma coordenação do trabalho faz-se necessária. Dois fatores são essenciais no controle de versão: a freqüência de atualizações e a quantidade de texto a ser atualizada.
18. **Permitir Avaliar os Índices de Aprendizagem, como**:
    a. **Tempo:**

i. Tempo total de percurso no modulo
ii. Tempo médio de permanência nas páginas do modulo
iii. Tempo médio de permanência por tipo de página (conceitos, exemplos ou exercícios)
iv. Tempo médio de leitura por tipo de pagina

b. **Percurso**:
v. Total de páginas acessadas no modulo. Estas páginas devem apresentar uma identificação que possibilite saber quais páginas foram acessadas.
vi. Total de retornos simples, volta a página anterior ou seqüência anterior.

c. **Movimento na navegação**:
vii. Paradas:
1. Paradas Bruscas: Trocas rápidas de páginas durante uma seqüência.
2. Paradas Grandes: Pelo tempo de permanência na pagina.
viii. Aceleração e desaceleração: Significando facilidade ou dificuldade de assimilar o conteúdo, respectivamente. Pode ser verificado pelo tempo de permanência.
ix. Mudanças bruscas de níveis: (conteúdo, exemplos e exercícios). Ex: O aluno está lendo um conteúdo e bruscamente vai para um exercício e começa a acertar todos os exercícios.
x. Blocos: Como foi a navegação por blocos como bloco de exercícios, bloco de exemplos, bloco de conteúdo, etc.
xi. Retroações: Quando o aluno se depara com uma dificuldade em uma página “X” em função da interferência devido a um não entendimento das páginas anteriores que o impede de prosseguir e que acaba fazendo-o retomar as páginas anteriores.
xii. *Loopings*: Fazer idas e voltas numa seqüência determinada.
xiii. Repetições de Scripts: Conjunto de procedimentos repetidos e “empacados” num domínio especifico de ações cognitivas.
xiv. Esquemas mentais: São verificados pelos conjuntos organizados de seqüências de navegação.

5.2 Capacidade de Cooperação

A cooperação na execução de tarefas apoiada por sistemas computacionais é uma necessidade crescente do mundo moderno. A cada minuto que passa novos processos estão sendo automatizados e realizados com o auxilio do computador. O que antes era uma tarefa fácil, hoje é uma questão de difícil solução, a percepção do ambiente de trabalho e das atividades realizadas pelos demais integrantes de um grupo de trabalho

que tem como objetivo realizar uma tarefa de forma cooperativa em um ambiente virtual.

A cooperação exige percepção e comunicação entre os participantes, exige formas de possibilitar que uma tarefa seja realizada por mais de um membro do grupo, assim os membros precisam saber o que precisa ser feito, o que já foi feito, qual a próxima tarefa, quem são os responsáveis pela tarefa, quem precisa de ajuda, quem está atrasado, etc. Os membros precisam de meios que os permitam auxiliar uns aos outros.

Dentro deste contexto o atual trabalho pretende obter uma estrutura de *workflow* que apresente capacidade de percepção, ferramentas que propiciem a troca de informações entre os participantes e regras de controle para possibilitar a cooperação.

Desta forma pode-se utilizar a coordenação, comunicação, percepção e memória de grupo para possibilitar a interação e necessita-se de procedimentos e protocolos para se obter o entendimento e convergência dos pontos de vista dos integrantes para realmente obtermos uma cooperação efetiva e eficiente.

5.3 Compatibilidade do Modelo

Objetivando desenvolver um modelo que seja compatível com o modelo de referência da WfMC e que possua todas as características mencionadas anteriormente, o desenvolvimento do modelo a que este trabalho se propõe, toma como base o modelo atual da WfMC ampliando e alterando onde for necessário para obter as características necessárias. O modelo da WfMC pode ser visto detalhadamente em [WOR 2002].

5.3.1 Meta-Modelo

O Meta-Modelo descreve as entidades contidas no nível mais elevado da Definição de Processo do *Workflow*, seus relacionamentos e atributos, conforme pode ser visto abaixo:

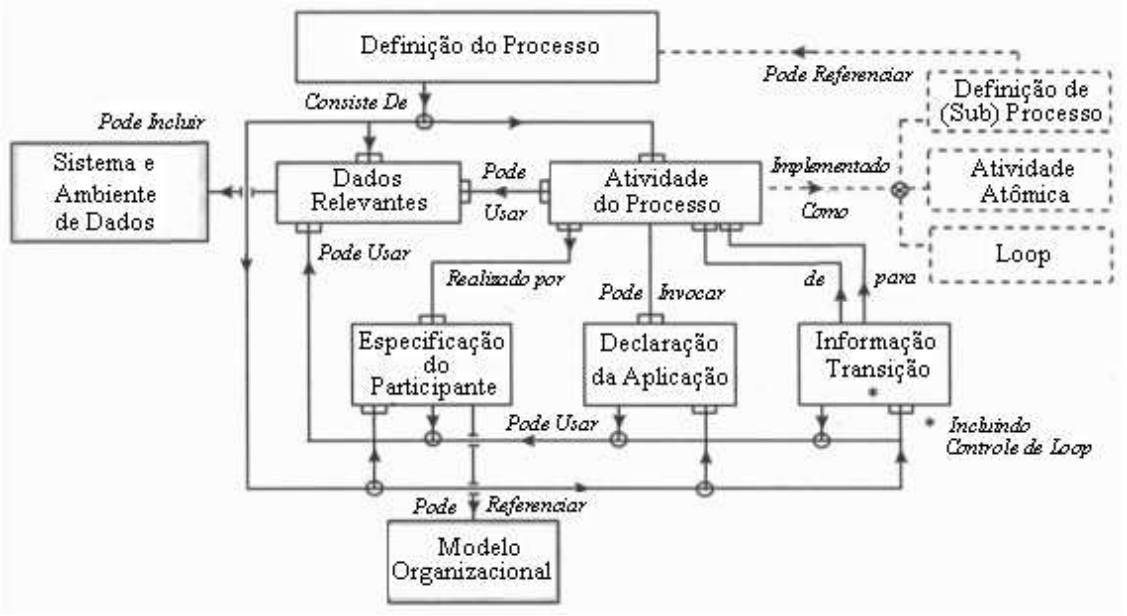


FIGURA 5.1 Entidades do Meta-Modelo

Para cada uma das entidades do meta-modelo, há um conjunto de atributos associados, alguns obrigatórios e outros opcionais, com os quais são descritas as características da entidade.

Neste ponto é interessante ressaltar que a WfMC incorporou no seu modelo, atributos extendidos, com a finalidade de quando necessário sejam utilizados para descrever características adicionais não previstas no modelo. Este atributos extendidos permitem uma extensão do modelo de forma controlada, o que nos permite utilizar este recurso para alcançar o objetivo deste trabalho.

5.3.2 Entidade de Definição de Processo de *Workflow*

A Definição de Processo de *Workflow* é definido pela WfMC como sendo a representação de um processo em uma forma que suporte manipulação automatizada, modelagem ou representação por um sistema de gerência de *workflow*. A Definição de Processo consiste de uma rede de atividades e seus relacionamentos, critérios para indicar o início e o termino do processo e informação sobre as atividades individuais, como participantes, aplicações associadas e dados, etc. [WOR 2002]

A entidade de definição de processo de *workflow* define os elementos que fazem parte do *workflow*. Contém as definições ou declarações, respectivamente, de atividades, transições, aplicações e de dados relevantes do processo. Os atributos desta entidade armazenam dados sobre a sua identificação, criação, simulação, responsabilidade e direito de acesso. Descreve o processo e provê estas informações a todas as demais entidades ao longo do *workflow*.

5.3.3 Atividades do Processo de *Workflow*

É utilizada para definir cada uma das atividades elementares que fazem parte do processo de *workflow*. Esta entidade possui atributos que armazenam as informações sobre o controle da atividade, as suas alternativas de implementação, a forma como será identificado o participante que realizará a atividade, os dados utilizados durante o processamento da atividade (como a sua prioridade, por exemplo) e os dados utilizados para a simulação de processos.

Neste modelo uma atividade pode ter a função de um conector, chamadas de atividades de roteamento, ou assumir outros quatro tipos possíveis de implementação: Aplicação, Sub-fluxo, Loop ou Sem Implementação. A WfMC define também atividades do tipo Roteamento. Através delas são implementados alguns tipos de conexões entre atividades.

5.3.4 Modelo Organizacional

Em cenários mais sofisticados a declaração dos participantes pode referir-se a um Modelo Organizacional, externo a definição de processo do *workflow*, o qual possibilita a avaliação de expressões mais complexas, incluindo referência a funções de empresariais e entidades organizacionais e seus relacionamentos.

5.3.5 Declaração de Aplicações do Workflow

Descreve as aplicações de TI que podem ser invocadas pelo serviço de *workflow* para dar suporte ao processamento associado com cada atividade. Tais aplicações podem ser ferramentas industriais genéricas ou qualquer outra disponível no ambiente.

5.3.6 Tipos Relevantes para o *Workflow*

Seção onde são definidas as declarações dos tipos de dados relevantes para o *workflow*, tipos simples ou complexos que podem ser utilizados na definição do *workflow*.

5.3.7 Dados Relevantes para o *Workflow*

Representam as variáveis do processo de *workflow* ou do modelo de definição de *workflow*. Estes atributos normalmente são utilizados para armazenar informações, dados sobre decisões tomadas ou parâmetros que precisam ser passados entre as diversas atividades do processo.

5.3.8 Sistema e Ambiente dos Dados

Dados mantidos pelo sistema de gerênciamento de *workflow*, mas podem ser acessados pelas atividades do *workflow* ou pelo sistema de gerência de *workflow* para avaliar expressões condicionais da mesma forma com que utiliza os dados relevantes do *workflow*. Sendo assim estes dados podem ser considerados uma extensão dos dados relevantes do *workflow*. Um pequeno Número de entidades de dados padronizadas estão definidos, mas outros tipos podem ser adicionados através de mecanismos de funções extendidas.

5.3.9 Tipos de Dados e Expressões

O meta-modelo, e o XPDL associado assumem um Número de tipos de dados padrão como string, reference, integer, float, date/time e etc, tais tipos de dados são relevantes para o *workflow*.

5.3.10 Modelo de *Workflow*, Definição e Repositório do Processo

Estas entidades vistas até o momento com seus respectivos atributos, suportam um mecanismo comum de descrição de definição de processo, constituindo assim o Mínimo Modelo de Processo, o qual pode ser expandido através do uso de atributos extendidos ou funções extendidas. O meta-modelo assume o uso de um repositório comum de definição de processo, associado com o sistema de gerência de *workflow*, para que com isso seja evitado a redefinição de definição de processos, como podemos ver abaixo:

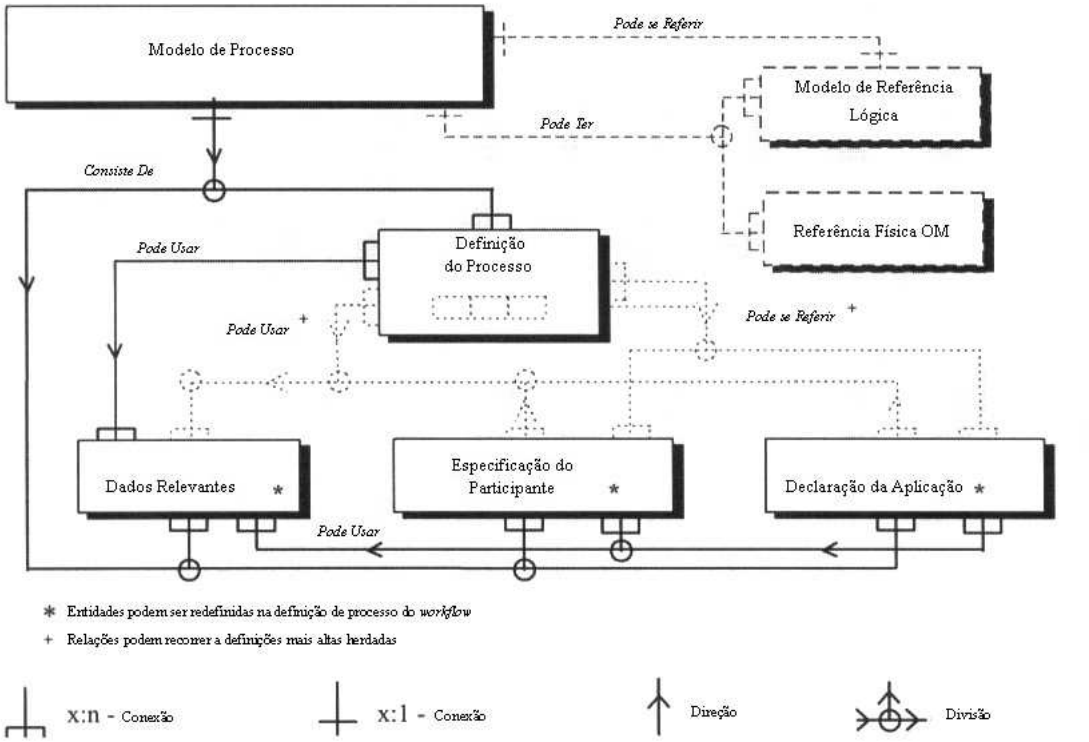


FIGURA 5.2 Entidades do Modelo de Processo do *Workflow*

Podemos ver na figura 6.3 a estrutura global da interface de importação e exportação de definição de processo e seus relacionamentos como o serviço de w*orkflow* associado.

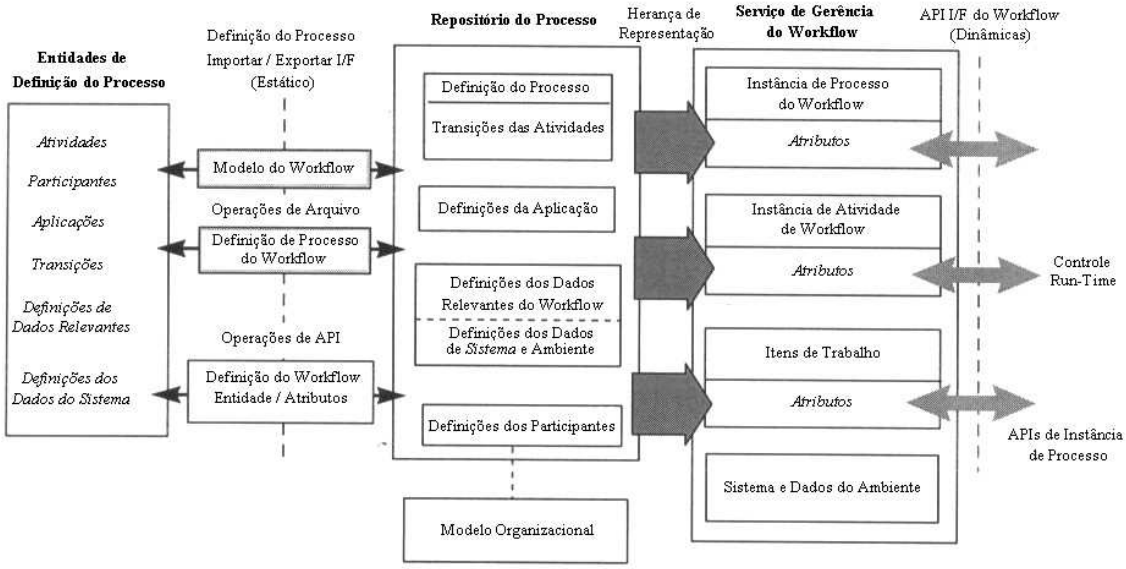


FIGURA 5.3 Interface de Importação e Exportação de Definição de Processo

5.3.11 Atributos

Na seqüência temos um resumo das entidades e atributos definidos no XPDL para o *Workflow*, são eles:

- Entidades:
  a) Definição do Modelo
  b) Definição do Processo
  c) Atividade do Processo
  d) Informação da Transição
  e) Declaração da Aplicação
  f) Dados Relevantes
  g) Definição dos Participantes
- Atributos:
  - Id, presente em (a, b, c, d, e, f, g).
  - Name, presente em (a, b, c, d, e, f, g).
  - Description, presente em (a, b, c, d, e, f, g).
  - XPDL-Version, presente em (a).
  - Source Vendor ID, presente em (a).
  - Creation Date, presente em (a, b).
  - Version, presente em (a, b).
  - Author, presente em (a, b).
  - *Characterset*, presente em (a, b).
  - *Codepage*, presente em (a, b).
  - *Country Key*, presente em (a, b).
  - Publication Status, presente em (a, b).
  - Extended Library, presente em (a, b).
  - Conformance Class, presente em (a).
  - Responsible, presente em (a, b).
  - External Model Ref, presente em (a).
  - Access Restriction, presente em (a, b, c).
  - Documentation, presente em (a, b, c).
  - Icon, presente em (a, b, c).
  - Priority Unit, presente em (a, b, c).
  - Cost Unit, presente em (a, b, c, g).
  - Extended Atribute, presente em (a, b, c, d, e, f, g).
  - Valid from Date, presente em (b).
  - Valid To Date, presente em (b).
  - Classification, presente em (b).
  - Parameters, presente em (b, e).
  - Duration Unit, presente em (b).
  - Duration, presente em (b, c).
  - Waiting Time, presente em (b, c).
  - Working Time, presente em (b, c).
  - Characteristic, presente em (c).
  - Automation Mode, presente em (c).
  - Split, presente em (c).

- Join, presente em (c).
- Loop, presente em (c).
- Inline Block, presente em (c).
- Participant Assignment, presente em (c).
- Implementation, presente em (c).
- Aplication Assignment, presente em (c).
- Instantiation, presente em (c).
- Condition, presente em (d).
- From, presente em (d).
- To, presente em (d).
- Tool Name, presente em (e).
- Parameters, presente em (e).
- Type, presente em (f, g).
- Value, presente em (f).
- Type Related Information, presente em (g).
- Strategy, presente em (g).
- Capacity, presente em (g).
- Prepare time, presente em (g).

Os atributos podem ser divididos em diferentes grupos:

- Todas as entidades tem atributos *id*, *name* e *description* em comum.
- O segundo grupo contem atributos específicos que caracterizam a respectiva entidade.
- Parâmetros, condições e valores referentes ao Processo/Modelo do *Workflow* e Dados Relevantes podem ser utilizados em expressões.
- O quarto grupo contém atributos que referenciam outras entidades
- Documentação e atributos contem informação a ser usada pela máquina de execução.
- O sexto grupo contem informação relevante para a simulação e otimização do processo.
- Atributos extendidos podem ser definidos para todas as entidades.

A WfMC faz um levantamento periódico dos atributos extendidos de forma a padronizar novos atributos que estejam sendo necessário para manter a conformidade do modelo.

5.3.12 Atributos Extendidos

Como foi visto anteriormente, todas as entidades podem possuir atributos extendidos, isto para que o modelo da WfMC possa atender à todas as necessidades de uma empresa qualquer. A WfMC aconselha que sejam utilizados dentro do possível apenas os atributos padronizados, porém se for absolutamente necessário podem ser incorporados quantos atributos extendidos forem necessários. Da mesma forma podem

ser utilizadas funções extendidas, fornecidas por um programa de qualquer origem que esteja instalado no ambiente.

Os atributos extendidos devem ser descritos conforme a tabela a seguir:

TABELA 6.2 Atributo Extendido

Nome do Atributo	**M/O**	**PalavraChave WPDL**	**Tipo de Dado**	**Descrição**
Identifier	M	EXTENDED_ATTRIBUTE	IDENTIFIER	Usado para identificar o atributo extendido
Attribute type	M		IDENTIFIER	Tipo de dado, tipos validos são: simples e complexo
Attribute value	M		Um valor do tipo correto	Valor necessário para a execução da tarefa
Attribute description	O		STRING	Uma descrição textual do atributo

O seu construtor em WPDL é da seguinte forma:

```
<extended attribute list>   ::=  EXTENDED_ATTRIBUTE
                                 <attribute id>
                                 <attribute type>
                                 <attribute value>
                                 [<description>]
                                 [<extended attribute list>]
<attribute id>              ::=  <identifier>
<attribute type>            ::=  <complex data type>
<attribute value>     ::=   <initial>          // of matching type
                            | <function acess>     // of matching result type
<description>               ::=  <string>
```

Sendo o seu construtor em XPDL:

```
<xsd:element name=”ExtendedAttribute”>
      <xsd:complexType mixed=”True”>
            <xsd:choice minOccurs=”0” maxOccurs=”unbounded”>
                  <xsd:any processContents=”lax” minOccurs=”0”
                  maxOccurs=”unbounded”/>
            </xsd:choice>
      <xsd:attribute name=”Name” type=”xsd:NMTOKEN”
          use=”required”/>
            <xsd:attribute name=”Value” type=”xsd:string”/>
   </xsd:complexType>
</xsd:element>

<xsd:element name=”ExtendedAttributes”>
      <xsd:complextype>
            <xsd:sequence>
                  <xsd:element ref=”xpdl:ExtendedAttribute”
                  minOccurs=”0” maxOccurs=”unbounded”/>
            </xsd:sequence>
      </xsd:complexType>
<xsd:element>
```

5.3.13 Controle de Fluxo Extendido

O meta-modelo da WfMC não especifica nenhum detalhe sobre o controle de fluxo interno para uma atividade, que esta especificada na seqüência de trabalho ou quanto as aplicações invocadas, ferramentas ou procedimentos os quais podem ser gerados com a atividade. Um atributo de controle de fluxo extendido pode ser definido pelo usuário oferecendo duas alternativas de esquemas de invocação nestas circunstancias de forma parelela ou seqüencial. Qualquer opção futura, tais como a inclusão de lógica condicional, é definida pelo usuário como um atributo extendido.

5.3.14 Bibliotecas Extendidas

Funções e procedimentos fornecidos pelo usuário podem ser adicionadas declarando uma livraria extendida. Isto permite acesso a dados no local do gerênciador do *workflow*. Definições de bibliotecas extendidas estão contidas na definição de processo ou onde aplicável em um grupo de definições de processo, dentro de um modelo de definição de processos. Funções e procedimentos são implementados localmente dentro do ambiente de execução do *workflow* e a identificação apropriada é declarada dentro da definição de bibliotecas extendidas.

Os atributos das bibliotecas extendidas podem ser usados no processo do *workflow* a na entidade do modelo do *workflow*. Isto permite declarações de bibliotecas de funções e procedimentos. Isto pode conter duas partes, declaração de funções e procedimentos, os quais incluirão atributos extendidos, veja na tabela abaixo:

TABELA 6.3 Atributo de Bibliotecas

Nome do Atributo	**M/O**	**WPDL Keyword**	**Data Type**	**Descrição**
Library Element Identifier	M			Identifica o elemento
		FUNCTION	Identifier	Identifica uma função da biblioteca
		PROCEDURE	Identifier	Identifica uma procedure da biblioteca
Result Type	M	RESULT	*	Um tipo de dado para denotar o tipo de resultado, somente para função de biblioteca
Name	O	NAME	String	Texto. Usado para identificar o elemento da biblioteca
Description	O	DESCRIPTION	String	Descrição textual Curta do elemento da biblioteca
Parameters	O			Parâmetros que são passados através de uma função ou procedure de uma biblioteca.

5.4 Banco de Dados

O modelo de *workflow* da WfMC conforme está publicado em [WOR 2002], não faz inferência a organização no banco, tão pouco restringe a um SGBD especifico. Esta sendo realizado na UFGRS, um trabalho de mapeamento do modelo da WfMC para um banco de dados em uma dissertação de mestrado que deve ser concluída em Dez/2003, o trabalho é do mestrando Montgmery Barroso [FRA 2003], orientando do professor José Valdeni de Lima.

O mapeamento foi feito baseando-se nas definições das entidades e atributos definidos pela WfMC, utilizando banco de dados em MySQL. Neste trabalho o banco em MySQL foi convertido para SQLServer 2000, por apresentar maior segurança quanto ao controle de integridade referencial e possuir uma interface mais amigável e melhor dominada por mim. O banco resultante foi adaptado conforme as necessidades durante o desenvolvimento.

Assim o presente trabalho é desenvolvido em Visual Basic 6.0, por ser uma linguagem bem aceita pelo mercado e também por que não estou apto a realizar este trabalho em outra linguagem de programação, também foram utilizadas as tecnologias de documentos HTML, ASP e SQL Server 2000.

O banco resultante da conversão de MySQL para SQLServer inicialmente consistia em 81 tabelas, as quais apresentam relacionamentos entre suas chaves primarias e estrangeiras para garantir a integridade do banco. O diagrama entidade-relacionamento completo, disposto em camadas de acordo com a seqüência de alimentação dos dados, respeitando a integridade referencial, assim seguindo de cima para baixo, vê-se claramente as tabelas dispostas em linhas, representando o nível em que se encontra, sendo necessário alimentar sempre as tabelas do nível anterior para então seguir com a alimentação dos dados nas tabelas abaixo.

Como é básico para a compreensão deste trabalho o conhecimento do banco de dados, vejamos agora detalhadamente as entidades e relacionamentos que o compõe:

No banco chamado de EnsinoOnLine devido ao direcionamento que este trabalho apresenta, as entidade podem ser dispostas de acordo com o fluxo dos dados, seguindo a integridade referencial. As regras de integridade referencial também apresentam a opção de exclusão em cascata onde necessária de forma que ao se excluir um registro da tabela inicial “Package” serão excluídos automaticamente todos os registros relacionados. As regras de integridade garantem não só a alimentação correta das entidades, como também impede que seja excluído algum registro incorretamente.

Assim considerando esta denominação virtual de “camadas” com o objetivo de determinar a seqüência dos dados de acordo com as restrições de integridade, podemos enumerar as 81 entidades provenientes do modelo da WfMC mapeado por Montgomery Barroso [FRA 2003], da seguinte forma:

- Nível 1:
  - GraphConformance
- Nível 2:
  - Package
- Nível 3:
  - ExternalPackage
  - Script
  - ExtendedAttribute_Package
  - Responsible

- Nivel 4:
  - ExtendedAttribute_ExternalPackage
  - Rel_Package_Responsible
- Nivel 5:
  - Element
- Nivel 6:
  - DataType_Type
  - Attribute
- Nivel 7:
  - DataType
- Nivel 8:
  - DataType_DeclaredType
  - DataType_ArrayType
  - DataType_BasicType
  - DataType_ExternalReferenceType
  - DataType_UnionType
  - DataType_ListType
  - DataType_EnumerationType
  - DataType_SchemaType
  - DataType_RecordType
- Nivel 9:
  - TypeDeclaration
  - DeclaredType
  - ArrayType
  - BasicType
  - ExternalReferenceType
  - UnionType
  - ListType
  - EnumerationValue
  - SchemaType
  - RecordType
- Nivel 10:
  - ExtendedAttribute_TypeDeclaration
- Nivel 11:
  - AccessLevel
  - DurationUnit
  - PublicationStatus
- Nivel 12:
  - WorkflowProcess
- Nivel 13:
  - Boolean
  - FormalParameter_Mode
- Nivel 14:
  - DataField
  - ExtendedAttribute_WorkflowProcess
  - FormalParameter_WorkflowProcess
- Nivel 15:
  - ExtendedAttribute_DataField
  - ParticipantType

    - ApplicationChoice
- Nivel 16:
    - Participant
    - Application
- Nivel 17:
    - Extendedattribute_Participant
    - FormalParameter_Application
- Nivel 18:
    - ApplicationChoice_ExternalReference
    - ExtendedAttribute_Application
    - ApplicationChoice_FormalParameter
- Nivel 19:
    - ActivitySet
    - Application_ExternalReference
    - Application_FormalParameter
- Nivel 20:
    - ActivityType_BlockActivity
- Nivel 21:
    - BlockActivity
- Nivel 22:
    - StartFinishMode
    - InstantiationType
    - ActivityType
- Nivel 23:
    - Activity
- Nivel 24:
    - Join_Split_Type
    - ActivityType_Implementation
    - ImplementationType
    - ExtendedAttribute_Activity
    - DeadLine
- Nivel 25:
    - TransitionRestriction
    - Implementation
- Nivel 26:
    - Split
    - ConditionType
    - Tool_Type
    - Implementation_Tool
    - Implementation_Subflow
    - ExecutionMode
    - ActualParameter_Subflow
- Nivel 27:
    - Transition
    - Tool
    - Subflow
- Nivel 28:
    - ExtendedAttibute_Transition
    - ExtendedAttribute_Tool

- ActualParameter_Tool

O detalhamento dos relacionamentos entre as entidades pode ser visto no anexo 1 onde temos o *script* de criação do banco de dados.

5.5 Estudo de Caso

O estudo de caso será uma simulação de execução de provas. Considerando que este trabalho está direcionado para ensino, decidiu-se pela experimentação de execuções de provas. As provas além de oferecerem a possibilidade de aplicar os recursos mencionados nos parágrafos anteriores, oferecem também a possibilidade de desenvolver e demonstrar uma flexibilidade de escolha quanto ao tipo de execução das provas, as quais podem ser:

- Quanto ao executor:
  - Individual
  - Equipe
- Quanto a ordem de execução:
  - Sequencial
  - Paralelo

Assim temos que uma prova pode ser definida como sendo de execução individual, isto é, executada por apenas um estudante, ou pode ser em equipe. As provas independentemente de serem individuais ou em equipe podem ter suas questões resolvidas na sequência em que foram apresentadas ou resolvidas na sequência que o aluno desejar. Estas opções devem ser definidas pelo autor do curso através de parâmetros no *workflow*, informados através de atributos extendidos.

5.6 Comentários Finais

Neste capítulo foi apresentado o modelo da WfMC, que foi tomado como base para o desenvolvimento deste trabalho. Sendo assim, este capítulo apresenta a modelagem utilizada onde podemos ver detalhadamente a estrutura adotada. O modelo foi mapeado para um banco de dados SQL Server com regras de restrição quanto à alimentação dos dados, impedindo que exclusões indevidas sejam realizadas e também proporcionando o mecanismo de exclusões em cascatas nos casos em que se faz necessário, tudo para garantir a integridade dos dados.

Sobre este banco foi construído o protótipo que apresenta a solução ao problema proposto. No capítulo 7 temos passo a passo a apresentação do protótipo.

6 Construção do Protótipo

O protótipo foi desenvolvido na linguagem Visual Basic 6.0, poderia ter sido desenvolvido em php, C, ou outra qualquer e o chamei de EnsinoOnLine, devido ao seu objetivo neste momento. Assim o protótipo é composto de um ambiente visual desenvolvido em Visual Basic 6.0 e alguns documentos HTML e ASP, que são visualizados dentro do próprio ambiente. Foram implementadas as funcionalidades necessárias para conferir ao sistema as características de cooperação e percepção propostas.

Considerando o estudo dos temas ligados aos aspectos de awareness e cooperação, podemos concluir que a percepção de grupo é dada pelo fornecimento de diversas informações relativas aos demais participantes do grupo, conforme foi visto no capítulo 3, enquanto que a cooperação é fruto das características de awareness somadas ao compartilhamento de informações, que conforme seja o caso deve ser apoiado por um ambiente e regras de integridade adequado, confome foi visto no capítulo 2 item 4.

Desta forma a construção do protótipo foi direcionada para a implementação de um ambiente com os recursos necessários, vislumbrados na análise da proposta deste trabalho e informações obtidas nas referencias bibliográficas.

Porém o modelo da WfMC não contempla estas informações mais detalhadas dos participantes. Informações como login, senha, e-mail, preferências, etc, histórico das atividades, não são necessariamente informações pertinentes ao modelo e podem ser atribuídas à máquina de *workflow*, considerando-se a meta deste trabalho de alterar o mínimo possível o modelo da WfMC mantendo assim a interoperabilidade deste trabalho com outros usuários do modelo da WfMC, optou-se por adicionar novas tabelas, com os atributos necessários, constituindo estas um novo conjunto de dados, atribuídos a máquina de *workflow*. Estas novas tabelas tem o nome genérico de EntMaq_nomedatabela, assim o prefixo EntMaq significa “Entidade da Máquina de *Workflow*”, difierenciando-a das demais tabelas provenientes do mapeamento do modelo da WfMC.

6.1 Aquisição do *Workflow*

Como foi visto anteriormente, o modelo de *workflow* utilizado neste trabalho é o modelo da WfMC, o qual foi mapeado para um banco de dados MySQL e posteriormente convertido para SQL Server. O mestrando Montgomery Barroso [FRA 2003], que fez o mapeamento, criou um procedimento de importação do *workflow* a partir de um aquivo texto escrito em XPDL. Este pequeno programa lê os atributos conforme estão descritos no arquivo em XPDL e insere os respectivos registros no banco, conforme pode ser visto na figura 6.3. A geração do arquivo em XPDL é feita por um outro módulo gráfico desenvolvido na dissertação do mestrando Thiago Telecken [TEL 2003], o qual foi implementado como uma opção do Amaya, figuras 6.1 e 6.2.

O procedimento inicia com a utilização do módulo gráfico desenvolvido pelo Thiago Telecken [TEL 2003], onde o usuário tem a sua disposição símbolos gráficos de

representação de *workflow*, baseados nos símbolos do modelo CASATI, tendo a possibilidade de configurar os atributos de cada figura que corresponde a informação sobre as tarefas, participantes, processo, etc. Após completamente alimentado, gerando uma definição gráfica como o anexo 2, o sistema gera um arquivo em texto escrito em XPDL, anexo 3, o qual é posteriormente lido pelo aplicativo do Montgomery [FRA 2003], que alimenta o banco de dados MySQL, anexo 4, e alimenta as tabelas respectivas no banco SQL Server.

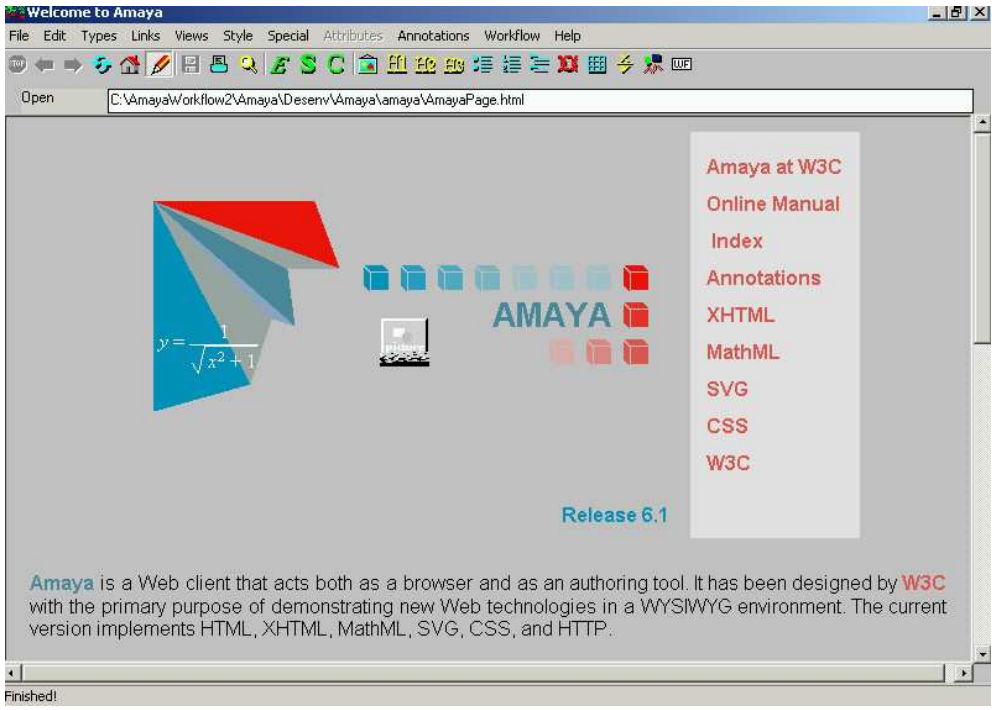


FIGURA 6.1 Amaya

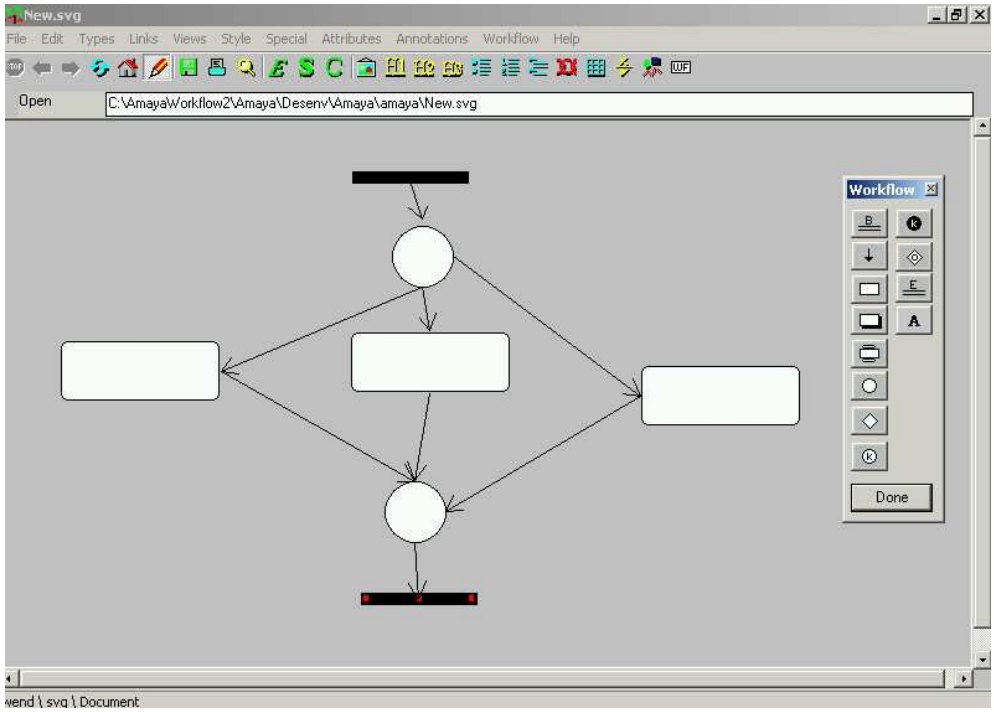


FIGURA 6.2 Novo workflow no Amaya

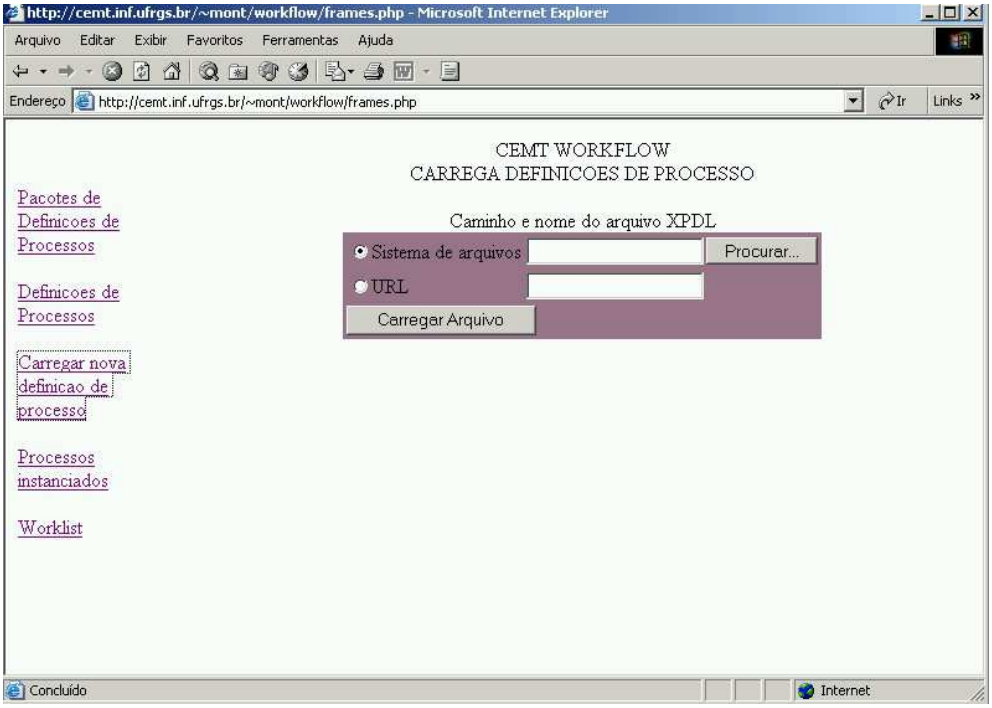


FIGURA 6.3 Leitura do arquivo em XPDL

6.2 Login e Cadastro de Novos Participantes

O monitoramento, o controle das ações dos participantes e a comunicação de cada um com o grupo são a base da percepção e posteriormente da cooperação, logo é muito importante que o sistema exija o login ao entrar no sistema, bem como possibilite o cadastro de todas as informações necessárias referente aos participantes. Caso o participante ainda não tenha cadastro ele pode cadastrar-se. O sistema apresenta as alternativas disponíveis, como cursos ou equipes de trabalho, ficando a escolha do participante o que deseja fazer. Ao mesmo tempo disponibiliza-o para ser convidado por um grupo que já esteja desenvolvendo um trabalho e ainda oferecendo-lhe as informações necessárias para que o comece a executar as tarefas que lhe são atribuídas. Todas estas opções conferem ao sistema mais autonomia, flexibilidade e capacidade de percepção e cooperação.

O modelo da WfMC não contempla as informações cadastrais do participante como login, senha, e-mail ou preferências, sendo assim foi definida uma tabela de usuários chamada *EntMaq_Usuários*, contendo os seguintes atributos:

- **Id**: Número inteiro, autonumeração, identificador unívoco da cada usuário do sistema.
- **Type**: Conforme definido em [WOR 2002] para os tipos de participantes do *workflow*, e encontrados no banco de dados na tabela *ParticipantType*, os tipos podem ser:
    - **RESOURCE_SET:** Um conjunto de recursos.
    - **RESOURCE:** Um recurso especifico.

  - **ROLE:** Este tipo permite uma atribuição de papel. Um papel neste contexo é uma função. Uma função não é necessariamente o papel de um participante, o coodenador pode para propósitos administrativos definir um conjunto de participantes ou recursos do sistema como um papel.
  - **ORGANIZATIONAL_UNIT:** Um departamento ou qualquer outra unidade com um modelo organizacional.
  - **HUMAN:** Um humano, que irá interagir com o sistema via sua interface.
  - **SYSTEM:** Um agente automático.
- **Nome**: Nome do usuário.
- **Login**: Login do usuário.
- **Senha**: Senha de identificação do usuário.
- **Email**: E-mail do usuário para envio de correspondências, este campo permite a inclusão de vários endereços separados por ponto-e-virgula de forma que para usuários que desejem receber seus e-mails com copias para várias contas o possam fazer.
- **Cidade**: Útil na questão de awareness percebendo de onde são os participantes do grupo. Possibilitando consultas de verificação de quantos participantes são de determinada cidade e ainda possibilita que o sistema possa oferecer ao participante os nomes dos participantes de sua cidade, ajudando-o a escolher um grupo de trabalho, e ainda oferecendo aos grupos de trabalho existentes a opção de um novo integrante, sempre tendo em vista a percepção e a colaboração.
- **Estado**: O estado onde está o participante, vale para este atributo o que foi dito para o atributo Cidade.
- **País**: Idem
- **Fone**: Número de telefone do participante, possibilitando uma forma a mais de comunicação.
- **Formação**: Objetiva permitir aos participantes terem uma noção do nível de compreensão de determinados assuntos, item importante na colaboração.
- **Outros**: Demais informações e observações que forem necessárias.

Como a entrada de dados se dá pela importação dos dados do XPDL, a qual não alimenta a tabela *entMaq_Usuários*, foi criado um gatilho na tabela *Participant* que verifica se o participante já existe na tabela *entMaq_Usuários*, se não existir o gatilho insere um novo registro tendo como login e senha do novo usuário o Id inserido na *Participant*. Abaixo temos o gatilho:

```
CREATE TRIGGER InsereEntMaqUsuario ON  [dbo].[Participant]
FOR INSERT
AS

declare
        @login varchar(18),
        @Id varchar(18),
        @Nome varchar(25)

set @login = ' '

select @Id = [Id], @Nome = [Name] from inserted
```

```
SELECT DISTINCT @login = { fn UCASE(Login) }
FROM        EntMaq_Usuarios
WHERE     ({ fn UCASE(Login) } = { fn UCASE(@Id) })

if not @login  >  ' '
  begin
          INSERT INTO EntMaq_Usuarios
                        (Nome, Login, Senha, email, Cidade, Estado, Pais, Fone, Formacao, Obs, DataNascimento)
          VALUES     ('seu nome', @Id, @Id, 'email@dominio', 'Cidade', 'Estado', 'Pais', 'Fone', 'Formação', 'Obs',
                              { fn    now() })
  end
```

6.3 Atividades Atômicas e Não Atômicas

Contemplando a possibilidade de agrupar quando necessário 2 ou mais atividades com o objetivo de obter-se um bloco de atividades afins, que devem ser executadas por um ou mais participantes com ou sem cooperação, o modelo apresenta as entidades de Activity, ActivitySet e BlockActivity, que possibilitam tratar uma atividade atomicamente, bem como um conjunto de atividades.

6.4 Atividades Programadas

O controle feito pela máquina de workflow sobre os prazos estipulados nas atividades, gera uma agenda de mensagens programadas, enviado-as aos responsáveis pelas tarefas quando for o momento. O quando deve ser enviada a mensagem é parametrizado. O parâmetro é qual o intevalo de tempo de antecedência que se deseja enviar a mensagem. Para isto o módulo de Agenda deve ficar rodando no servidor, 24 horas por dia, para que seja possível monitorar a programação e enviar as mensagens programadas. Para possibilitar a programação de mensagens que não estejam previstas nas atividades do workflow, mas que seja necessário agenda-las, como por exemplo um lembrete, foi adicionado ao módulo de Agenda a opção de incluir agendamento de mensagens. As mensagens sempre são entregues de duas formas: Via e-mail e por janelas de mensagens na interface do programa, sendo a mensagem via e-mail enviada pelo servidor na data pré-determinada e a mensagem na janela de mensagens da interface do programa será enviada no promeiro momento em que o usuário fizer login após a data predeterminada.

Para estas funcionalidades foram criadas duas novas tabelas EntMaq_Parametros e EntMaq_Mensagens.

Na EntMaq_Parametros encontramos o atributo “Periodo_Antecedencia”, no qual é setado o intervalo de tempo com o qual se deseja que sejam enviadas as mensagens. Na EntMaq_Mensagens encontramos os seguintes atributos:

- **Id**: Número inteiro, autonumeração.
- **De_Usuário**: Id do usuário que está enviando a mensagem.
- **Para_Usuário**: Id do usuário que deve receber a mensagem.
- **Titulo**: Título da mensagem.

- **Mensagem**: Texto da mensagem.
- **Data_Envio**: Data a ser enviada a mensagem.
- **Modo_Email**: Um campo bit, assumindo os valores 0 ou 1. Valor 0 (zero) para não enviada, e valor 1 (um) para mensagem enviada.
- **Modo_Interface**: Um campo bit, assumindo os valores 0 ou 1. Valor 0 (zero) para não enviada, e valor 1 (um) para mensagem enviada.

O sistema possui uma opção de consulta para visualizar a lista de todas as mensagens programadas enviadas ou não.

6.5 Visualização do Estado do Sistema

O estado das diversas atividades poderá ser visto de forma gráfica através do feedback dado pelo EnsinoOnLine na forma de um arquivo texto escrito em WDPL, contendo a informação sobre o estado de todas as entidades, utilizando-se do atributo extendido previsto no modelo da WfMC, sendo este arquivo lido pela interface gráfica do Tiago Telecken o qual irá atualizar o modelo gráfico criado originalmente no início do processo, tendo agora a informação de como está o andamento das tarefas. Esta visualização na interface do Thiago Telecken ainda não está implementada, ficando assim apenas a opção de geração do arquivo, a qual pode ser automatizada para ser realizada automaticamente a cada intervalo de tempo escolhido pelo administrador do sistema.

O estado das diversas atividades também pode ser visto em caixas de listagem e relatórios criados com está finalidade, disponíveis em todas as telas e opções do sistema, fazendo parte dos requisitos de awareness e cooperação. Embora não sejam expressas de forma gráfica sobre o modelo, são expressas de forma inteligível, através de uma consulta individual referindo-se a uma atividade especifica, ou participante especifico, ou uma consulta de múltiplos registros como a consulta de todas as atividades de um processo especifico ou de todos os processos em execução. Sendo o estado controlado e definido conforme o modelo da WfMC, ficando a cargo da máquina de *workflow* apenas gerênciar as atualizações deste atributo.

Os estados possíveis para processos e atividades foram determinados pela WfMC, conforme está descrito em [WOR 99]. O presente trabalho adotou estes estados, sem nenhuma alteração.

A WfMC define os seguintes estados e sub-estados:

- Aberto.
  - Rodando.
  - Não Rodando.
    - Não Iniciado.
    - Suspenso.
- Fechado
  - Abortado
  - Terminado
  - Completado

Os estados podem ser expresso apenas no seu último nível como por exemplo: Suspenso, ou indicando todos os níveis, separando-os com ponto ".", no mesmo caso ficaria Aberto.Não Rodando.Suspenso. A figura 7.1 abaixo mostra as transições para os vários estados:

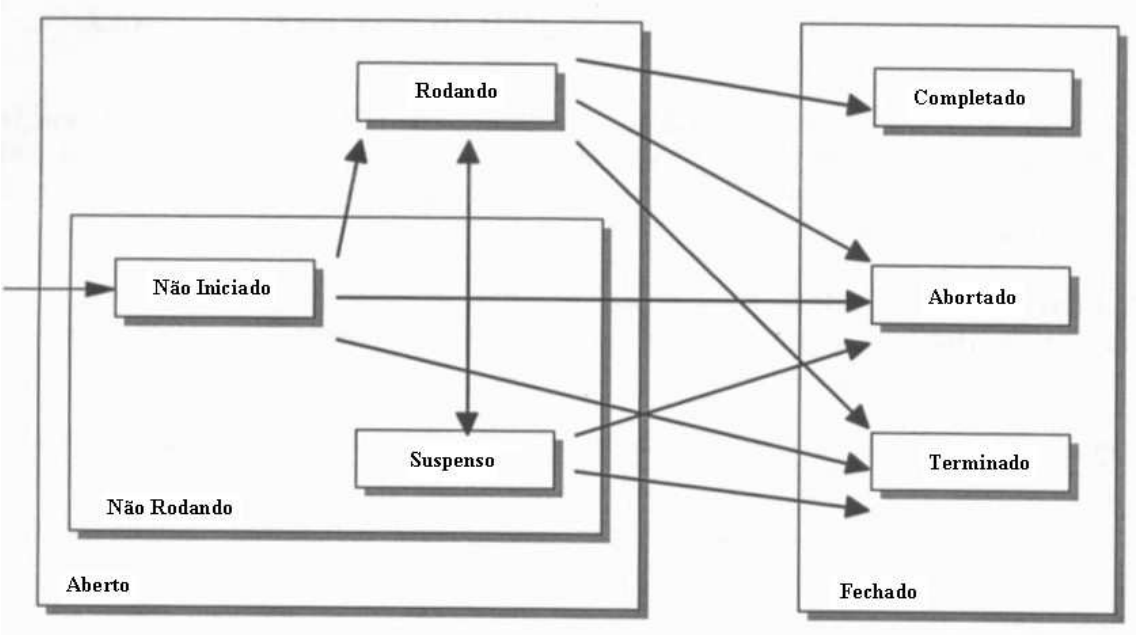


FIGURA 6.4 Transições entre estados

A priori no modelo apresentado por [FRA 2003], não existem atributos específicos para o estado das atividades. Desta forma a maquina de workflow precisa controlar os estados das atividades e de todo o processo em uma entidade própria para este fim, exibindo o estado das atividades conforme o ambiente e a opção desejada. No caso deste trabalho o ambiente EnsinoOnLine desenvolvido apresenta através de caixas de listagens e relatório a situação do estado de cada atividade e exporta um documento XPDL com um atributo Status contendo a informação sobre o estado de cada atividade. Este documento XPDL pode ser lido por exemplo, pelo Amaya utilizado para criar inicialmente o workflow e baseado na informação de Status pode apresentar de forma gráfica no seu editor SVG o Status de cada atividade, processo e pacote como um todo. A implementação desta visualização gráfica no editor Amaya não é parte deste trabalho, sendo assim deverá ser implementada posteriormente.

Assim a entidade criada para controlar o Status das atividades foi denominada entMaq_Status, contendo os seguintes atributos:

6.6 Comunicação

As opções de comunicação implementadas foram o envio de e-mail, as mensagens em estilo Tickertape e sessões fechadas em estilo chat, onde os participantes podem formar grupos fechados para discutir um assunto comum a todos, onde podem tomar decisões ou simplesmente discutir sobre um assunto.

Para a implementação dos grupos fechados de chat, foram criadas duas novas tabelas. A primeira tabela, EntMaq_Conferencia, contém o cadastro da Conferência, com os seguintes atributos:

- **Id**: Número inteiro autonumerador.
- **Titulo**: Título indicativo do assunto ou objetivo da conferência.
- **Responsavel**: Id do usuário que propôs abertura do grupo de conferência.
- **Data_Inicio**: Data de abertura da conferência.
- **Data_Fim**: Data de finalização da conferência.

A segunda tabela, EntMaq_Conferencia_Participante, contém os participantes convidados para formar o grupo de discussão, composta pelos seguintes atributos:

- **Conferencia_Id**: Id identificador da conferência, proveniente da EntMaq_Conferencia.
- **Participante_Id**: Id do usuário convidado a fazer parte do grupo.
- **Data_Inicio**: Data em que o usuário foi convidado a fazer parte do grupo.
- **Data_Fim**: Data em que o usuário saiu da discussão.

Os participantes não convidados, podem solicitar a sua inclusão no grupo, fazendo o pedido ao responsável pelo grupo, sendo o único que pode fazer a inclusão de novos integrantes no grupo.

6.7 Awareness

Como foi visto no capítulo 3, awareness ou percepção é um estado mental, induzido pelas informações de contexto. Sendo assim o sistema desenvolvido atende bem aos requisitos necessários para proporcionar este estado mental aos participantes.

A implementação da caixa de mensagem estilo “Tickertape”, a possibilidade de chat fechados como conferência, o envio de e-mail, agendamento de mensagens, mensagens enviadas pelo sistema quando os prazos estão vencendo, avisos sobre usuários (logados, logando-se ou desconectando-se), cadastro dos usuários (localização, idade, conhecimentos), possibilidade de executar as tarefas sendo auxiliado pelos demais integrantes do grupo, e uma ampla gama de consultas sobre o sistema, proporciona ao participante uma visão geral do ambiente, que em alguns casos pode ser melhor do que teria em um trabalho em grupo onde todos estivessem fisicamente no mesmo local.

6.8 Cooperação

O trabalho cooperativo conforme foi visto no item 2.4 é o resultado da percepção somada a disponibilização de resultados intermediários, como documentos de um repositório ou ainda da simples troca de informações em alguma de comunicação entre

os participantes. Sendo possível diferenciarmos 4 tipos de comportamento cooperativo [LAB 98], sendo que dentre eles, foi adotado apenas um tipo de comportamento cooperativo, o de Assistência, onde a tarefa é alocada para somente um agente no grupo e os outros podem ajudar este agente se for necessário.

Para isso foi implementada a cooperação obtida pela troca de informações por diversas formas de comunicação e controles disponíveis no sistema, e quando solicitado pelos participantes, compartilhando a execução das tarefas. No aspecto de troca de informações foi implementada uma versão adaptada do Tickertape [PAR 98], chats e e-mail, enquanto no aspecto de compartilhamento da execução das tarefas, foi implementada através da disponibilização da visualização dos trabalhos desenvolvidos por um determinado participante que está realizando a tarefa, sendo possível que vários participantes visualizem as informações, mas apenas 1 esteja em um determinado momento editando. Assim muitos podem colaborar na execução da tarefa, sendo que o sistema permite que a qualquer momento um outro usuário possa requerer a função de edição, ficando todos os demais apenas com a opção de visualização. Qualquer tarefa pode ser executada desta forma, bastando para isso que o executor responsável pela tarefa, abilite-a através de uma opção na barra de menu, como tarefa cooperativa. A partir deste momento o sistema passa a gerênciar as visualizações e o direito de edição, através de comandos na barra de menu e caixas de listagem informando quem está visualizando a tarefa, permitindo a troca de informações e a troca do direito de opção. Novamente para restringir alterações no modelo, a disponibilidade de realização da cooperação é intriseca ao sistema, permitindo que qualquer tarefa possa ser realizada cooperativamente, ou ainda que seja realizada cooperativamente por um lapso de tempo apenas e logo volte a ser realizada individualmente. Para controlar este processo o sistema conta com uma nova tabela da máquina, chamada EntMaq_Cooperação, a qual possui os seguintes atributos:

- **Id_Reg**: Um Número inteiro, autonumeração, identificando univocamente o registro.
- **Package_Id**: Id do pacote ao qual pertence a atividade, dado proveniente da tabela Activity.
- **Id**: Id da atividade dentro do Pacote, dado proveniente da tabela Activity.
- **Responsável_Id:** Id do responsável por editar naquele momento a atividade, dentro do contexto de cooperação, isto quer dizer que não necessariamente este responsável é o mesmo ao qual foi atribuída a tarefa, código proveniente da tabela EntMaq_Usuário, inserido no momento em que um usuário requer o direito de edição.
- **Editando**: Um campo tipo bit, que assume os valores 0 (zero) ou 1. 0 (Zero) quando naquele momento as ultimas alterações foram salvas e 1 quando ainda não foram salvas as ultimas alterações. Isto se faz necessário para que no momento em que um participante requeira o direito de edição daquela tarefa, o sistema apenas aceite a substituição após as alterações que estão sendo feitas pelo atual participante sejam salvas.
- **Ativo**: Um campo bit, que controla os períodos em que a tarefa está no modo de execução cooperativa ou está no modo de execução individual. Assim em um dado momento, ao surgirem dificuldades o responsável pela atividade pode passa-la ao estado (Ativo = 1), onde o sistema então permitira a sua visualização pelos demais participante e ainda que outro participante a edite, permitindo que após solucionado um determinado problema de forma cooperativa a tarefa retorne a execução individual (Ativo = 0), até a sua finalização.

- **Data_Inicial**: A data e a hora em que a tarefa passou a condição de tarefa realizada por cooperação.
- **Data_Final**: A data e a hora em que a tarefa passou a condição de tarefa realizada de forma individual, em conjutno com a Data_Inicial, permiti traçar um histórico da realização das tarefas no tocante a forma pela qual foram realizadas.

6.9 Histórico

O histórico do sistema é proporcionado por consultas e relatórios feito sobre a base de dados, utilizando as tabelas já mencionadas, e também uma tabela específica de histórico onde os eventos são registrados como um log do sistema. Neste histórico são registrados todos os eventos que ocorrem, sendo esta tabela a origem de controles e dos principais indicadores de tempo, atrasos e seqüência de realização das tarefas.

A tabela de histórico chamada EntMaq_Histórico, contém os seguintes atributos:

- **Id:** Número inteiro autonumerção.
- **Package_Id:** Id do pacote.
- **WorkflowProcess_Id:** Id do processo.
- **Application_Id:** Id da aplicacao.
- **ActivitySet_Id:** Id do conjunto de atividades.
- **Activity_Id:** Id da atividade.
- **Participant_Id:** Id do participante.
- **Usuario_Id:** Id do usuário.
- **Data_Ocorrência:** Data do evento.
- **Tipo:** O tipo de evento, que pode ser:
  - Inserir
  - Atualizar
  - Excluir
  - Consultar
  - Entrar
  - Sair
  - Enviar
  - Receber
  - Suspender
  - Terminar
  - Concluir
  - Abortar
  - Abrir
  - Fechar
  - Ler
  - Escrever
  - Responder
  - Perguntar
- **Modulo:** Módulo em que o evento foi gerado.

- **Comando:** Comando interno que foi executado, facilitando rastrear exatamente a conseqüência de determinadas ações.
- **Observacoes:** Texto que o pode ser inserido quando necesario.

6.10 Sistema de Coordenação

O sistema de coordenação dos trabalhos implementado é constituído de uma interface onde o administrador pode consultar o estado das tarefas, o histórico das atividades, enviar mensagens, agendar mensagens e consultar qualquer dado do sistema.

Com base nestas informações o coordenador dos trabalhos pode gerênciar adequadamente o sistema, sendo-lhe proporcionado ferramentas necessária para obter informações e atuar no sistema.

6.11 Arquitetura do Sistema

O sistema baseia-se em uma estrutura cliente-servidor (figura 6.6). O SGBD utilizado foi o SQL SERVER 2000, o banco de dados criado possui as tabelas de dados do modelo do workflow definidas no trabalho do Montgomery Barroso França [FRA 2003], as tabelas definidas neste trabalho como tabelas da máquina de workflow e as tabelas de instâncias de dados do objeto do workflow, o que no case apresentado é constituído pelas provas a serem aplicadas.

Desta forma temos na figura 6.5 os diagrama esquemático representando a arquitetura do sistema proposto. O sistema utiliza uma característica do Amaya, inserida no trabalho do mestrando Tiago Telecken, que lhe possibilita a edição gráfica de workflow e exportar o workflow definido graficamente para um arquivo no formato xpdl. Este arquivo é então lido pelo compilador desenvolvido no trabalho do mestrando Montgomery Barroso França [FRA 2003], o qual traduz os conceitos do xpdl para comandos de transaction SQL, gerando um script de alimentação do banco. Este script alimenta as informações relativas ao modelo do workflow, sendo que após isto a maquina de workflow processa estes dados alimentando as informações relativas a maquina de workflow e aos dados das instâncias das atividades desenvolvidas.

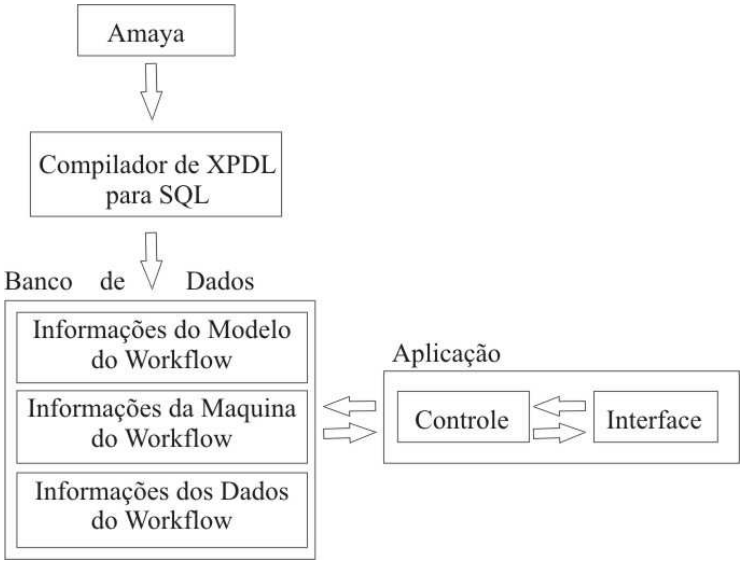


FIGURA 6.5 Arquitetura do Sistema

Desta forma cada cliente atua individualmente através da interface da aplicação, e a camada de controle faz a ligação com o banco de dados no servidor, filtrando as informações solicitadas por um processo ou usuário e inserindo informações relativas aos eventos e resolução das atividades.

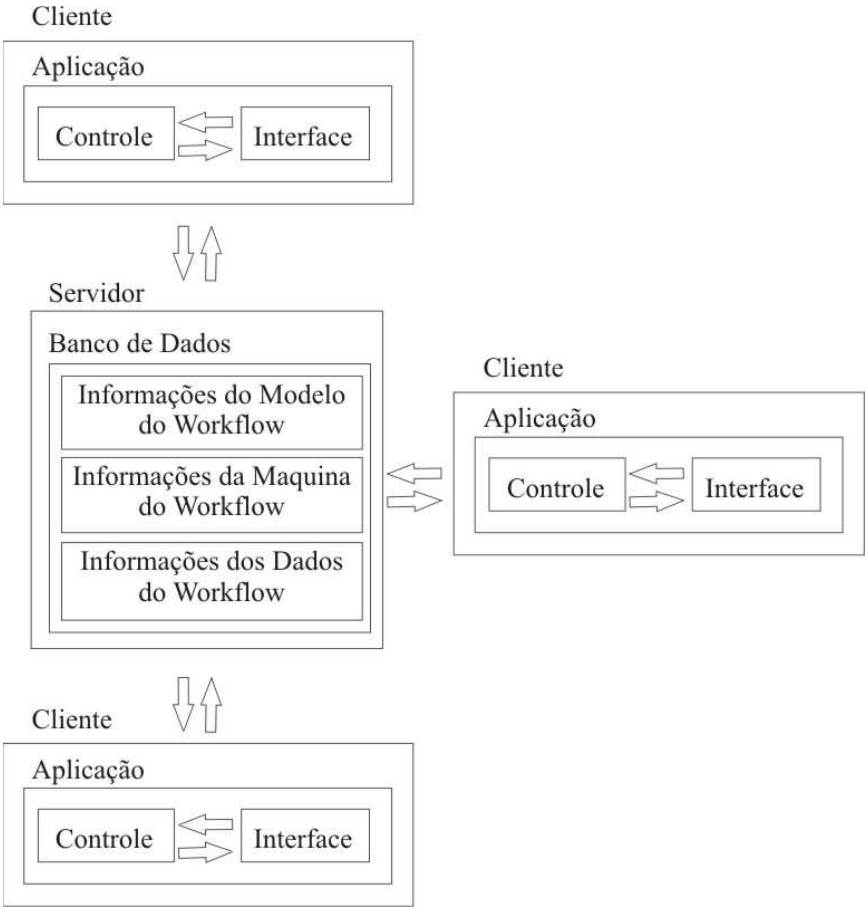


FIGURA 6.6 Diagrama Cliente-Servidor

6.12 Estudo de Caso

Esta dissertação tem como objetivo contribuir com o projeto CEMT, embora os resultados obtidos possam ser aplicados a qualquer ambiente, o ambiente ao qual este trabalho esta dirigido é o de ensino. O sistema foi desenvolvido para aplicação inicial em uma rede local que com pequenas alterações pode ser aplicado via internet.

Considerando as necessidades em um contexto onde os educadores precisam disponibilizar aos seus educandos um ambiente de aprendizagem munido de ferramentas de controle de atividades, controle de pendências, percepção do ambiente e possibilidade de cooperação na execução das atividades, o estudo de caso foi feito considerando um curso de HTML com aplicação de provas. Assim centralizando a atenção na aplicação das provas. Incluindo no modelo dois parâmetros com as informações sobre a forma de aplicar e como será executada a prova.

Estes parâmetros foram inseridos como atributos extendidos, os quais receberam a denominação de FormaAplicacao e FormaExecucao.

Desta forma ao gerar o XPDL do workflow devem ser incluídas na atividade, informações através de atributos extendidos, conforme segue:

- FormaAplicacao
  - Individual

    - Grupo
- FormaExecução
    - Seqüencial
    - Paralela

A máquina de *workflow* ao analisar a informação destes parâmetros, configura a forma como será executada e apresentada a prova.

Possibilitando desta forma que o educador possa definir um determinado conjunto de questões como sendo uma prova a ser resolvida em equipe, outro conjunto de questões será uma prova a ser resolvida individualmente. Temos ainda a situação de que o educador precise aplicar um conjunto de questões como sendo uma prova que deverá ser resolvida na seqüência apresentada ou não, desta forma o educador pode optar dentro de um conjunto de possibilidades apresentado. Agora na questão de quem são os educandos que devem responder a prova, a quem a prova deve ser aplicada? Qual é a composição da equipe no caso de aplicação da prova a ser resolvida em equipe?

A prova será uma atividade no conjunto de atividades que compõe um curso, logo faz parte de um Package, assim o sistema pode aplicar a prova a todos os participantes de um determinado curso e através de mais um parâmetro que indique o número de integrantes de cada grupo, no caso de provas a serem aplicadas em grupo, o sistema pode deixar opcional a formação dos grupos, abrindo a possibilidade através da interação e percepção a formação das equipes como acontece normalmente em um ambiente físico onde estão todos os educandos reunidos. O sistema apresenta como pendência uma prova em equipe e oferece a opção para que cada participante do curso convide um ou mais colegas para formarem um grupo para a execução da prova, a limitação será dada pelo parâmetro NumIntegrantesGrupo, a ser fornecido pelo educador através de um atributo extendido.

6.12.1 Seqüência da Aplicação

O uso da aplicação iniciou-se com a descrição do processo em uma modelagem, modelo de *workflow* da WfMC, através da modificação incluída no Amaya em [TEL 2003]. Logo após o modelo exportado para XPDL pelo Amaya foi compilado para um script, contendo as instruções de transaction SQL, que alimentam o banco de dados com a definição do modelo e com os parametros sobre a forma de execução das provas e o número de integrantes de cada grupo. Foi definido um curso de HTML contendo diversos capítulos e quatro provas, uma para cada tipo de situação.

O script gerado pelo compilador [FRA 2003], foi executado no Query Analyser do Microsoft SQL Server 2000, alimentando o banco de dados com as informações oriundas da modelagem.

Após esta etapa inicial, o sistema estava pronto para receber a inscrição de novos alunos. Através da tela inicial de login, é possível acessar um formulário de inscrição de novos integrantes, que após a inscrição tem acesso ao sistema desenvolvido com todas as suas implementações de apoio ao trabalho, com percepção, cooperação e controle de tarefas.

Uma vez logado no sistema, o usuário é recebido com uma série de informações sobre a sua situação de trabalho, sobre o ambiente e sobre as opções disponíveis de ações a serem tomadas.

Neste case o usuário após o efetuar o seu logon recebe a informação sobre os cursos disponíveis para os quais ele pode se registrar. Uma vez selecionado o curso a máquina de workflow lhe apresenta a seqüência de atividades a serem executadas, os demais integrantes inscritos no mesmo curso e quem esta logado no mesmo instante. Sendo possível trocar mensagens instantâneas através do *tickertape* ou ainda por e-mail. Conforme o usuário vai executando as tarefas que lhe foram designadas o sistema vai registrando as atividades e controlando os prazos, uma vez que ocorra algum atraso o sistema notifica ao educador ou toma as ações previstas para o atraso em questão. Ao chegar o momento da prova a maquina de *workflow* apresenta-a conforme estabelecido nos parametros passado através dos atributos extendidos. No caso da prova em equipe o sistema apresenta uma tela onde é possível convidar outros integrantes do curso, dentro do limite estabelecido pelo educador, e ainda na resolução da prova os participantes de um grupo podem trocar informações entre si e trocar o direito de edição da prova, realizando a cooperação na resolução da prova.

6.13 Considerações Finais

Neste capítulo foi visto o desenvolvimento do sistema, o qual foi direcionado a sanar cada uma das necessidades levantadas na proposta. Desta forma o sistema desenvolvido é o resultado da soma de muitas ferramentas direcionadas à obtenção do estado mental de percepção do ambiente pelos integrantes, somadas à capacidade de cooperação, o que no caso deste trabalho é a cooperação por assistência, baseada na troca de informações e a utilização do modelo da WfMC, sem alterações.

Pode-se perceber que soluções simples, agrupadas de forma coerente podem resultar em uma solução complexa e útil, como esta que foi desenvolvida.

Na descrição de cada item deste capitulo esta a solução adotada para cada objetivo proposto.

7 Conclusão

A questão inicial a que este trabalho se propôs a resolver foi a obtenção de um modelo de workflow com capacidade de percepção de eventos. No decorrer do desenvolvimento podemos concluir que percepção ou awareness, não é dada pelo modelo, mas sim pelo ambiente em que este modelo é utilizado. As informações necessárias à obtenção da percepção não necessariamente precisam estar incluídas no modelo, podendo estar armazenada e gerênciada no que podemos chamar de ambiente da máquina de workflow, como foi demonstrado por este trabalho.

A questão da realização de tarefas de forma cooperativa é uma função do tipo de tarefa e do tipo do ambiente em que elas são realizadas, como foi visto no item 2.4. Desta forma para o presente trabalho, devido às suas características e objetivos, foi abordado apenas a modalidade de cooperação por troca de informações, com visualização compartilhada e edição permitida a todos do grupo, porém um de cada vez.

Este trabalho reuniu o que há de melhor no estado da arte sobre modelagem de workflow, obtenção de awareness e trabalho cooperativo. Sendo o resultado da compilação deste conjunto de ferramentas, ajustado a um caso de treinamento, apresentando a etapa de aplicação de provas com algumas alterações, ou restrições de acordo com o aplicável.

Sendo um dos objetivos deste trabalho, a obtenção da percepção e colaboração aplicáveis ao modelo da WfMC em uma proposta de ensino, compatibilizando-o com os objetivos do projeto CEMT, e ainda utilizando os trabalhos do mestrando Tiago Telecken (editor gráfico) [TEL 2003] e Montgomery Barroso (mapeamento do modelo da WfMC para uma base de dados) [FRA 2003], integrantes do mesmo projeto. Este objetivo foi alcançado, pois se demonstrou neste trabalho uma aplicação que utiliza o modelo da WfMC, sem alterações, utilizando-se apenas dos atributos extendidos previstos no modelo. Isto através de uma máquina de workflow, desenvolvida com a finalidade de dar suporte a percepção e a colaboração, utilizando-se das mais variadas formas de auxilio ao usuário.

Como trabalhos futuros, vejo a necessidade de converter este sistema para programas livres, como o php, o linux, o mysql, e a compilação dos trabalhos do Thiago Telecken [TEL 2001] e Montgomery Barroso [FRA 2003] em uma só interface e ainda um estudo mais detalhado da interface 5 do WfMC, buscando as similaridades e possíveis melhorias em ambos os sistemas.

Temos ainda como trabalhos futuros o aprimoramento desta solução, de modo a torna-la mais flexível e adaptável na execução de workflows mais diversificados, partindo da proposta deste trabalho e desenvolvendo uma metodologia de trabalho mais modular e orientada a objeto de forma a tornar-se mais abrangente e adaptativa no quesito tipo de processos modelados.

Anexo 1 Script de Criação do Banco de Dados

```
create table PublicationStatus(
Cod integer primary key,
Status varchar(18));

/* Tabela de Enumeracao (Fixa) */
insert into PublicationStatus (Cod, Status) values (1,'UNDER_REVISION');
insert into PublicationStatus (Cod, Status) values (2,'RELEASED');
insert into PublicationStatus (Cod, Status) values (3,'UNDER_TEST');-- drop table GraphConformance;

create table GraphConformance(
Cod integer PRIMARY KEY,
Type varchar(20));

-- Tabela de enumeracao , fixa.
INSERT INTO GraphConformance (Cod,Type) VALUES (1,'FULL_BLOCKED');
INSERT INTO GraphConformance (Cod,Type) VALUES (2,'LOOP_BLOCKED');
INSERT INTO GraphConformance (Cod,Type) VALUES (3,'NON_BLOCKED');

create table Package(
Id varchar(18) primary key,
Name varchar(25),
XPDLVersion varchar(12) not null,
Vendor varchar(12) not null,
Created varchar(30) not null,
Description varchar(12),
Documentation varchar(12),
PriorityUnit varchar(12),
CostUnit varchar(12),
PublicationStatus INTEGER NULL,
Author varchar(12),
Version varchar(12),
Codepage varchar(12),
CountryKey varchar(12),
GraphConformance INTEGER NOT NULL,
-- INDEX (GraphConformance),
FOREIGN KEY (GraphConformance) REFERENCES GraphConformance(Cod)
ON DELETE NO ACTION,
-- INDEX (PublicationStatus),
FOREIGN KEY (PublicationStatus) REFERENCES PublicationStatus(Cod)
ON DELETE NO ACTION);

create table AccessLevel(
cod integer PRIMARY KEY,
Type varchar(15));

/* Enumeracao tabela fixa */
insert into AccessLevel (Cod,Type) values (1,'PUBLIC');
insert into AccessLevel (Cod,Type) values (2,'PRIVATE');
insert into AccessLevel (Cod,Type) values (3,'UNDEFINED');

create table DurationUnit(
cod integer PRIMARY KEY,
Type varchar(15));

/* Enumeracao tabela fixa */
insert into DurationUnit (Cod,Type) values (1,'YEAR');
insert into DurationUnit (Cod,Type) values (2,'MONTH');
insert into DurationUnit (Cod,Type) values (3,'DAY');
insert into DurationUnit (Cod,Type) values (4,'HOUR');
insert into DurationUnit (Cod,Type) values (5,'MINUTE');
insert into DurationUnit (Cod,Type) values (6,'SECOND');

create table WorkflowProcess(
Package_Id varchar(18) NOT NULL,
Id varchar(18)  NOT NULL,
Name varchar(25),
AccessLevel_Cod integer NOT NULL,
Created varchar(12) ,
Description varchar(12) ,
Priority varchar(12) ,
```

```
Limite varchar(12),
ValidFrom varchar(12),
ValidTo varchar(12),
WaitingTime varchar(12),
WorkingTime varchar(12),
Duration varchar(12),
DurationUnit_Cod integer NULL,
PublicationStatus INTEGER NULL,
Author varchar(12),
Version varchar(12),
Codepage varchar(12),
CountryKey varchar(12),
FirstActivity integer NULL,
PRIMARY KEY (Package_Id, Id),
-- INDEX chave_estrangeira1 (PublicationStatus),
FOREIGN KEY (PublicationStatus) REFERENCES PublicationStatus(Cod)
ON DELETE NO ACTION,
-- INDEX chave_estrangeira2 (DurationUnit_Cod),
FOREIGN KEY (DurationUnit_Cod ) REFERENCES DurationUnit (Cod)
ON DELETE NO ACTION,
-- INDEX chave_estrangeira3 (AccessLevel_Cod),
FOREIGN KEY (AccessLevel_Cod) REFERENCES AccessLevel(Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Package*/
-- INDEX chave_estrangeira4 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE);

create table Script(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Version varchar(18),
Grammar varchar(18),
Type varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod),
/* Eh entidade fraca em relacao ao Package e tambem relaciona-se a ele */
-- INDEX Chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE);

create table Responsible(
Responsible varchar(25) PRIMARY KEY);

create table Rel_Package_Responsible(
Package_Id varchar(18) NOT NULL,
Responsible varchar(25) NOT NULL,
PRIMARY KEY (Package_Id,Responsible),
/* Relaciona-se com Package */
-- INDEX Chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relaciona-se com Responsible */
-- INDEX chave_estrangeira2 (Responsible),
FOREIGN KEY (Responsible) REFERENCES Responsible(Responsible)
ON DELETE NO ACTION);

create table Element(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25),
Parent integer ,
Content varchar(50),
PRIMARY KEY (Package_Id,Cod),
/* Eh entidade fraca em relacao ao Package */
-- INDEX chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
-- INDEX chave (Cod),
/* Auto-relacionamento */
-- INDEX chave_estrangeira2 (Package_Id,Parent),
FOREIGN KEY (Package_Id,Parent) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION);

create table Attribute(
Package_Id varchar(18) NOT NULL,
Element_Cod integer NOT NULL,
```

```
Id varchar(18) NULL,
Value varchar(50) NOT NULL,
-- INDEX (Package_Id,Element_Cod),
FOREIGN KEY (Package_Id,Element_Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE CASCADE);

create table ExternalPackage(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Href varchar(25),
PRIMARY KEY (Package_Id,Cod) ,
/* Eh entidade fraca em relacao ao Package e tambem relaciona-se a ele*/
-- INDEX chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE);

create table ExtendedAttribute_ExternalPackage(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Cod integer NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com ExternalPackage */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Cod),
FOREIGN KEY (Package_Id,Parent_Cod) REFERENCES ExternalPackage(Package_Id,Cod)
ON DELETE CASCADE);

create table ExtendedAttribute_Package(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
PRIMARY KEY (Package_Id,Cod) ,
/* Relaciona-se com Package */
-- INDEX chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION);

create table DataType_Type(
Cod integer PRIMARY KEY,
Type varchar (18));

/* Tabela de Enumeracao (Fixa) */
insert into DataType_Type(Cod, Type ) values (1,'BasicType');
insert into DataType_Type(Cod, Type ) values (2,'DeclaredType');
insert into DataType_Type(Cod, Type ) values (3,'SchemaType');
insert into DataType_Type(Cod, Type ) values (4,'ExternalReference');
insert into DataType_Type(Cod, Type ) values (5,'RecordType');
insert into DataType_Type(Cod, Type ) values (6,'UnionType');
insert into DataType_Type(Cod, Type ) values (7,'ArrayType');
insert into DataType_Type(Cod, Type ) values (8,'ListType');
insert into DataType_Type(Cod, Type ) values (9,'EnumerationType');

create table DataType(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Type integer,       /* Especifica o tipo da especializacao */
MemberOf INTEGER NULL,
PRIMARY KEY (Package_Id,Cod),
UNIQUE (Package_Id,Cod,Type),
/* Eh entidade fraca em relacao ao Package */
-- INDEX chave_estrangeira (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
-- INDEX chave_estrangeira2 (Type),
FOREIGN KEY (Type) REFERENCES DataType_Type(Cod)
ON DELETE NO ACTION);
```

```
create table DataType_BasicType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_BasicType(Cod) values (1);

create table BasicType_Type(
Cod integer PRIMARY KEY,
Type varchar(12));

/* Tabela de Enumeracao (Fixa) */
insert into BasicType_Type(Cod,Type) values (1,'STRING');
insert into BasicType_Type(Cod,Type) values (2,'FLOAT');
insert into BasicType_Type(Cod,Type) values (3,'INTEGER');
insert into BasicType_Type(Cod,Type) values (4,'REFERENCE');
insert into BasicType_Type(Cod,Type) values (5,'DATETIME');
insert into BasicType_Type(Cod,Type) values (6,'BOOLEAN');
insert into BasicType_Type(Cod,Type) values (7,'PERFORMER');

create table BasicType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
Type integer,
/* Eh uma especializacao de DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 1 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_BasicType(Cod)
ON DELETE NO ACTION,
/* Valor enumerado */
-- INDEX chave_estrangeira3 (Type),
FOREIGN KEY (Type) REFERENCES BasicType_Type(Cod)
ON DELETE NO ACTION);

create table DataType_DeclaredType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_DeclaredType(Cod) values (2);

create table DeclaredType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 2 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_DeclaredType(Cod)
ON DELETE NO ACTION);

create table DataType_SchemaType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_SchemaType(Cod) values (3);

create table SchemaType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod,DataType_Type) REFERENCES DataType(Package_Id,Cod,Type)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 3 */
```

```
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_SchemaType(Cod)
ON DELETE NO ACTION,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira3 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION);

create table DataType_ExternalReferenceType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_ExternalReferenceType(Cod) values (4);

create table ExternalReferenceType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
Xref varchar(50),
Location varchar(50) NOT NULL,
Namespace varchar(50),
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod,DataType_Type) REFERENCES DataType(Package_Id,Cod,Type)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 4 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_ExternalReferenceType(Cod)
ON DELETE NO ACTION);

create table DataType_RecordType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_RecordType(Cod) values (5);

create table RecordType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod,DataType_Type) REFERENCES DataType(Package_Id,Cod,Type)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 5 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_RecordType(Cod)
ON DELETE NO ACTION);

create table DataType_UnionType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_UnionType(Cod) values (6);

create table UnionType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod,DataType_Type) REFERENCES DataType(Package_Id,Cod,Type)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 6 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_UnionType(Cod)
ON DELETE NO ACTION);

create table DataType_EnumerationType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_EnumerationType(Cod) values (9);
```

```
-- drop table EnumerationValue;
create table EnumerationValue(
Package_Id varchar(18) NOT NULL,
EnumerationType_Cod integer  NOT NULL,
Name varchar(50) NOT NULL,
DataType_Type INTEGER NOT NULL,
PRIMARY KEY (Package_Id,EnumerationType_Cod,Name),
/* Relacao com EnumerationType */
-- INDEX chave_estrangeira (Package_Id,EnumerationType_Cod,DataType_Type),
FOREIGN KEY (Package_Id,EnumerationType_Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 9 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_EnumerationType(Cod)
ON DELETE NO ACTION);

create table DataType_ArrayType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_ArrayType(Cod) values (7);

create table ArrayType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
LowerIndex integer,
UpperIndex integer,
DataType_Cod integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 7 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_ArrayType(Cod)
ON DELETE NO ACTION,
/* Relaciona-se com DataType  */
-- INDEX chave_estrangeira3 (Package_Id,DataType_Cod ),
FOREIGN KEY (Package_Id,DataType_Cod ) REFERENCES DataType(Package_Id,Cod)
ON DELETE NO ACTION);

create table DataType_ListType(
Cod integer PRIMARY KEY);

/* Tabela de Enumeracao (Fixa) */
insert into DataType_ListType(Cod) values (8);

create table ListType(
Package_Id varchar(18) NOT NULL,
Cod integer  NOT NULL,
PRIMARY KEY (Package_Id,Cod),
DataType_Type integer,
DataType_Cod integer,
/* Eh entidade fraca em relacao a DataType */
-- INDEX chave_estrangeira (Package_Id,Cod,DataType_Type),
FOREIGN KEY (Package_Id,Cod,DataType_Type) REFERENCES DataType(Package_Id,Cod,Type)
ON DELETE NO ACTION,
/* Restricao para que so possa ser do tipo 8 */
-- INDEX chave_estrangeira2 (DataType_Type),
FOREIGN KEY (DataType_Type) REFERENCES DataType_ListType(Cod)
ON DELETE NO ACTION,
/* Relaciona-se com DataType  */
-- INDEX chave_estrangeira3 (Package_Id,DataType_Cod ),
FOREIGN KEY (Package_Id,DataType_Cod ) REFERENCES DataType(Package_Id,Cod)
ON DELETE CASCADE);

-- INDEX (Package_Id,MemberOf);
ALTER TABLE DataType ADD FOREIGN KEY (Package_Id,MemberOf) REFERENCES RecordType(Package_Id,Cod)
ON DELETE NO ACTION;

-- INDEX (Package_Id,MemberOf);
ALTER TABLE DataType ADD FOREIGN KEY (Package_Id,MemberOf) REFERENCES UnionType(Package_Id,Cod)
ON DELETE NO ACTION;
```

```
create table TypeDeclaration(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
Name varchar(25),
Description varchar(12),
DataType_Cod INTEGER,
PRIMARY KEY (Package_Id,Id),
/* Eh entidade fraca em relacao ao Package e tambem relaciona-se a ele*/
-- INDEX chave_estrangeira (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relaciona-se com DataType */
-- INDEX chave_estrangeira2 (Package_Id,DataType_Cod),
FOREIGN KEY (Package_Id,DataType_Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE NO Action);

create table ExtendedAttribute_TypeDeclaration(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com TypeDeclaration */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES TypeDeclaration(Package_Id,Id)
ON DELETE CASCADE);

create table ParticipantType(
Cod integer PRIMARY KEY,
Type varchar(20));

/* Tabela de enumeracao fixa */
insert into ParticipantType (Cod,Type) values (1,'RESOURCE_SET');
insert into ParticipantType (Cod,Type) values (2,'RESOURCE');
insert into ParticipantType (Cod,Type) values (3,'ROLE');
insert into ParticipantType (Cod,Type) values (4,'ORGANIZATIONAL_UNIT');
insert into ParticipantType (Cod,Type) values (5,'HUMAN');
insert into ParticipantType (Cod,Type) values (6,'SYSTEM');

create table Participant(
Package_Id varchar(18) not null,
Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18),
Name varchar(25),
ParticipantType_Cod INTEGER NOT NULL,
Description varchar(50),
Xref varchar(50),
Location varchar(50),
Namespace varchar(50),
PRIMARY KEY (Package_Id,Id),
/* Relacao com Package*/
-- INDEX chave_estrangeira (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relacao com WorkflowProcess */
-- INDEX chave_estrangeira2 (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE NO ACTION,
/* Relacao com ParticipantType */
-- INDEX chave_estrangeira3 (ParticipantType_Cod),
FOREIGN KEY (ParticipantType_Cod) REFERENCES ParticipantType(Cod)
ON DELETE NO ACTION);

create table ApplicationChoice(
Cod integer PRIMARY KEY,
Choice varchar(20));

/* Tabela de enumeracao fixa */
insert into ApplicationChoice (Cod,Choice) values (1,'ExternalReference');
insert into ApplicationChoice (Cod,Choice) values (2,'FormalParameter');
```

```
create table ApplicationChoice_ExternalReference(
Cod integer PRIMARY KEY);

/* Tabela de enumeracao fixa */
insert into ApplicationChoice_ExternalReference (Cod) values (1);

create table ApplicationChoice_FormalParameter(
Cod integer PRIMARY KEY);

/* Tabela de enumeracao fixa */
insert into ApplicationChoice_FormalParameter (Cod) values (2);

create table Application (
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18),
Name varchar(25),
Description varchar(12),
ApplicationChoice_Cod integer NOT NULL,
PRIMARY KEY (Package_Id,Id),
UNIQUE (Package_Id,Id,ApplicationChoice_Cod),
/* Entidade fraca em relacao a Package */
-- INDEX chave_estrangeira (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relacao com WorkflowProcess */
-- INDEX chave_estrangeira2 (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE NO ACTION,
/* Especifica o tipo de aplicacao */
-- INDEX chave_estrangeira3 (ApplicationChoice_Cod),
FOREIGN KEY (ApplicationChoice_Cod) REFERENCES ApplicationChoice(Cod)
ON DELETE NO ACTION);

create table Application_ExternalReference(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
ApplicationChoice_cod integer NOT NULL DEFAULT 1,
Xref varchar(50),
Location varchar(50) ,
Namespace varchar(50),
PRIMARY KEY (Package_Id,Id),
/* Entidade fraca em relacao a Application*/
-- INDEX chave_estrangeira (Package_Id,Id,ApplicationChoice_Cod),
FOREIGN KEY (Package_Id,Id,ApplicationChoice_Cod ) REFERENCES Application(Package_Id,Id,ApplicationChoice_Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 1 */
-- INDEX chave_estrangeira2 (ApplicationChoice_cod),
FOREIGN KEY (ApplicationChoice_cod) REFERENCES ApplicationChoice_ExternalReference(Cod)
ON DELETE NO ACTION);

create table Application_FormalParameter(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
ApplicationChoice_Cod integer NOT NULL DEFAULT 2,
PRIMARY KEY (Package_Id,Id),
/* Entidade fraca em relacao a Application*/
-- INDEX chave_estrangeira (Package_Id,Id,ApplicationChoice_Cod),
FOREIGN KEY (Package_Id,Id,ApplicationChoice_Cod ) REFERENCES Application(Package_Id,Id,ApplicationChoice_Cod)
ON DELETE CASCADE,
/* Restricao para que so possa ser do tipo 2 */
-- INDEX chave_estrangeira2 (ApplicationChoice_cod),
FOREIGN KEY (ApplicationChoice_cod) REFERENCES ApplicationChoice_FormalParameter(Cod)
ON DELETE NO ACTION);

create table FormalParameter_Mode(
Cod integer PRIMARY KEY,
Mode varchar(20));

/* Tabela de enumeracao fixa */
insert into FormalParameter_Mode (Cod,Mode) values (1,'IN');
insert into FormalParameter_Mode (Cod,Mode) values (2,'OUT');
insert into FormalParameter_Mode (Cod,Mode) values (3,'INOUT');

create table FormalParameter_Application (
```

```
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
Indice varchar(18),
FormalParameter_Mode_Cod INTEGER NOT NULL,
DataType_Cod integer NOT NULL,
Application_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id, Id,Application_Id),
/* Relacionamento com DataType */
-- INDEX (Package_Id,DataType_Cod),
FOREIGN KEY (Package_Id,DataType_Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE NO ACTION,
-- INDEX (FormalParameter_Mode_Cod),
FOREIGN KEY (FormalParameter_Mode_Cod) REFERENCES FormalParameter_Mode(Cod)
ON DELETE NO ACTION,
/* Relacionamento com Application */
-- INDEX (Package_Id,Application_Id),
FOREIGN KEY (Package_Id,Application_Id) REFERENCES Application (Package_Id,Id)
ON DELETE CASCADE);

create table FormalParameter_WorkflowProcess (
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
Indice varchar(18),
FormalParameter_Mode_Cod INTEGER NOT NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
DataType_Cod integer NOT NULL,
PRIMARY KEY (Package_Id, Id,WorkflowProcess_Id),
/* Relacionamento com DataType */
-- INDEX (Package_Id,DataType_Cod),
FOREIGN KEY (Package_Id,DataType_Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE NO ACTION,
-- INDEX (FormalParameter_Mode_Cod),
FOREIGN KEY (FormalParameter_Mode_Cod) REFERENCES FormalParameter_Mode(Cod)
ON DELETE NO ACTION,
/* Relacionamento com WorkflowProcess */
-- INDEX (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE CASCADE);

create table ExtendedAttribute_Application(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Application */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES Application(Package_Id,Id)
ON DELETE CASCADE);

create table Boolean(
Cod integer PRIMARY KEY,
Boolean varchar(8));

/* Tabela fixa */
INSERT INTO Boolean (Cod,Boolean) values (1,'TRUE');
INSERT INTO Boolean (Cod,Boolean) values (2,'FALSE');

create table DataField(
Package_Id varchar(18) not null,
Cod INTEGER NOT NULL,
Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18),
Name varchar(25),
IsArray INTEGER DEFAULT 2,
Description varchar(12),
DataType_Cod integer NOT NULL,
InitialValue varchar(25),
Length varchar(25),
PRIMARY KEY (Package_Id,Cod),
UNIQUE (Package_Id,WorkflowProcess_Id,Id),
```

```
/* Relacao com Package*/
-- INDEX chave_estrangeira (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relacao com WorkflowProcess */
-- INDEX chave_estrangeira2 (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE NO ACTION,
/* Relacao com Boolean */
-- INDEX chave_estrangeira3 (IsArray),
FOREIGN KEY (IsArray) REFERENCES Boolean(Cod)
ON DELETE NO ACTION,
/* Relacao com DataType */
-- INDEX chave_estrangeira4 (Package_Id,DataType_Cod),
FOREIGN KEY (Package_Id,DataType_Cod) REFERENCES DataType(Package_Id,Cod)
ON DELETE NO ACTION);

create table ExtendedAttribute_DataField(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Cod INTEGER,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com DataField */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Cod),
FOREIGN KEY (Package_Id,Parent_Cod) REFERENCES DataField(Package_Id,Cod)
ON DELETE CASCADE);

create table ExtendedAttribute_WorkflowProcess(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira1 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Package*/
-- INDEX chave_estrangeira2 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relaciona-se com WorkflowProcess */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE NO ACTION);

create table ActivitySet(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
FirstActivity integer NULL,
PRIMARY KEY (Package_Id,Id),
UNIQUE (Package_Id, WorkflowProcess_Id, Id),
/* Relaciona-se com Workflow */
-- INDEX chave_estrangeira1 (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE CASCADE);

create table ActivityType(
Cod integer PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into ActivityType (Cod,Type) values (1,'ROUTE');
insert into ActivityType (Cod,Type) values (2,'IMPLEMENTATION');
insert into ActivityType (Cod,Type) values (3,'BLOCKACTIVITY');

create table ActivityType_Implementation(
Cod integer PRIMARY KEY);
```

```
/* Tabela de enumeracao. Fixa */
insert into ActivityType_Implementation (Cod) values (2);

create table ActivityType_BlockActivity(
Cod integer PRIMARY KEY);

/* Tabela de enumeracao. Fixa */
insert into ActivityType_BlockActivity (Cod) values (3);

create table InstantiationType(
Cod integer PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into InstantiationType (Cod,Type) values (1,'ONCE');
insert into InstantiationType (Cod,Type) values (2,'MULTIPLE');

create table StartFinishMode(
cod INTEGER PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into StartFinishMode (Cod,Type) values (1,'AUTOMATIC');
insert into StartFinishMode (Cod,Type) values (2,'MANUAL');

-- --------- ACTIVITY -----------------

create table Activity(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
ActivitySet_Id varchar(18) NULL,
Name varchar(25) NULL,
Description varchar(25),
Limite varchar(25),
ActivityType_Cod integer NOT NULL,
Performer varchar(18),
StartMode integer NULL,
FinishMode integer NULL,
Priority varchar(25),
Icon varchar(25),
Documentation varchar(25),
/* componentes de SimulationInformation */
Cost  varchar(25),
WaitingTime varchar(25),
WorkingTime varchar(25),
Duration varchar(25),
Instantiation_Cod integer NULL,
PRIMARY KEY (Package_Id,Id),
UNIQUE (Package_Id, Id, ActivityType_Cod),
UNIQUE (Package_Id,WorkflowProcess_Id,Id,ActivityType_Cod),
UNIQUE (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Id),
UNIQUE (Package_Id,WorkflowProcess_Id,Id),
-- INDEX (Package_Id,WorkflowProcess_Id,Id),
-- INDEX (Package_Id,Id,ActivityType_Cod),
-- INDEX (Package_Id,WorkflowProcess_Id,Id,ActivityType_Cod),
/* Relaciona-se com Workflow */
-- INDEX (Package_Id,WorkflowProcess_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id) REFERENCES WorkflowProcess(Package_Id,Id)
ON DELETE CASCADE,
/* Relaciona-se com ActivitySet (nao obrigatorio) */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id) REFERENCES
ActivitySet(Package_Id,WorkflowProcess_Id,Id)
ON DELETE NO ACTION,
/* Especifica o tipo da Especialidade */
-- INDEX chave_estrangeira3 (ActivityType_Cod),
FOREIGN KEY (ActivityType_Cod) REFERENCES ActivityType(Cod)
ON DELETE NO ACTION,
/* Especifica o modo de Start */
-- INDEX chave_estrangeira4 (StartMode),
FOREIGN KEY (StartMode) REFERENCES StartFinishMode(Cod)
ON DELETE NO ACTION,
/* Especifica o modo de Finish*/
-- INDEX chave_estrangeira5 (FinishMode),
```

```
FOREIGN KEY (FinishMode) REFERENCES StartFinishMode(Cod)
ON DELETE NO ACTION,
-- INDEX (Package_Id,Performer),
FOREIGN KEY (Package_Id,Performer) REFERENCES Participant(Package_Id,Id)
ON DELETE NO ACTION,
-- INDEX (Instantiation_Cod),
FOREIGN KEY (Instantiation_Cod) REFERENCES InstantiationType (Cod)
ON DELETE NO ACTION);

create table ExecutionMode(
Cod integer PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into ExecutionMode (Cod,Type) values (1,'ASYNCHR');
insert into ExecutionMode (Cod,Type) values (2,'SYNCHR');

create table Deadline(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
DeadlineCondition varchar(25) NOT NULL,
ExceptionName varchar(25) NOT NULL,
ExecutionMode_Cod integer NOT NULL,
Activity_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod),
/* Entidade fraca de Package */
-- INDEX chave_estrangeira1 (Package_Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relacao com Modo de Execucao */
-- INDEX chave_estrangeira2 (ExecutionMode_Cod),
FOREIGN KEY (ExecutionMode_Cod) REFERENCES ExecutionMode(Cod)
ON DELETE NO ACTION,
/* Relacao com Activity */
-- INDEX chave_estrangeira3 (Package_Id,Activity_Id),
FOREIGN KEY (Package_Id,Activity_Id) REFERENCES Activity(Package_Id,Id)
ON DELETE NO ACTION);

-- IMPLEMENTATION ------------------------------------

create table Implementation_Type(
Cod integer PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into Implementation_Type (Cod,Type) values (1,'NO');
insert into Implementation_Type (Cod,Type) values (2,'TOOL');
insert into Implementation_Type (Cod,Type) values (3,'SUBFLOW');

create table Implementation_Tool(
Cod integer PRIMARY KEY);

/* Tabela de enumeracao. Fixa */
insert into Implementation_Tool (Cod) values (2);

create table Implementation_Subflow(
Cod integer PRIMARY KEY);

/* Tabela de enumeracao. Fixa */
insert into Implementation_Subflow (Cod) values (3);

create table Implementation(
Package_Id varchar(18) NOT NULL,
Activity_Id varchar(18) NOT NULL,
ActivityType_Cod integer NOT NULL,
ImplementationType_Cod integer NOT NULL,
PRIMARY KEY (Package_Id,Activity_Id),
UNIQUE (Package_Id,Activity_Id,ImplementationType_Cod),
/* Especializacao de Activity */
-- INDEX (Package_Id,Activity_Id,ActivityType_Cod),
FOREIGN KEY (Package_Id, Activity_Id,ActivityType_Cod) REFERENCES Activity (Package_Id, Id,ActivityType_Cod)
ON DELETE CASCADE,
/* Fixa que o valor de ActivityType_Cod deve ser igual 2 */
-- INDEX (ActivityType_Cod),
FOREIGN KEY (ActivityType_Cod) REFERENCES ActivityType_Implementation(Cod)
ON DELETE NO ACTION,
```

```
/* Especifica o tipo de Implementation */
-- INDEX (ImplementationType_Cod),
FOREIGN KEY (ImplementationType_Cod) REFERENCES Implementation_Type(Cod)
ON DELETE NO ACTION);

create table Tool_Type(
Cod integer PRIMARY KEY,
Type varchar(18));

/* Tabela de enumeracao. Fixa */
insert into Tool_Type (Cod,Type) values (1,'APPLICATION');
insert into Tool_Type (Cod,Type) values (2,'PROCEDURE');

create table Tool(
Package_Id varchar(18) NOT NULL,
Activity_Id varchar(18) NOT NULL,
Description varchar(25),
ToolType_Cod integer NOT NULL,
ImplementationType_Cod integer NOT NULL,
Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Activity_Id),
/* Especializacao de Implementation */
-- INDEX chave_estrangeira1 (Package_Id,Activity_Id,ImplementationType_Cod),
FOREIGN KEY (Package_Id,Activity_Id,ImplementationType_Cod) REFERENCES Implementation
(Package_Id,Activity_Id,ImplementationType_Cod)
ON DELETE CASCADE,
/* Fixa que o valor de ImplementationType_Cod deve ser igual 2 */
-- INDEX chave_estrangeira2 (ImplementationType_Cod),
FOREIGN KEY (ImplementationType_Cod) REFERENCES Implementation_Tool(Cod)
ON DELETE NO ACTION,
/* Especifica o tipo de Tool */
-- INDEX chave_estrangeira3 (ToolType_Cod),
FOREIGN KEY (ToolType_Cod) REFERENCES Tool_Type(Cod)
ON DELETE NO ACTION);

create table Subflow(
Package_Id varchar(18) NOT NULL,
Activity_Id varchar(18) NOT NULL,
Description varchar(25),
ExecutionMode_Cod integer NOT NULL,
ImplementationType_Cod integer NOT NULL,
Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Activity_Id),
/* Especializacao de Implementation */
-- INDEX chave_estrangeira1 (Package_Id,Activity_Id,ImplementationType_Cod),
FOREIGN KEY (Package_Id,Activity_Id,ImplementationType_Cod) REFERENCES Implementation
(Package_Id,Activity_Id,ImplementationType_Cod)
ON DELETE CASCADE,
/* Fixa que o valor de ImplementationType_Cod deve ser igual 3 */
-- INDEX chave_estrangeira2 (ImplementationType_Cod),
FOREIGN KEY (ImplementationType_Cod) REFERENCES Implementation_Subflow(Cod)
ON DELETE NO ACTION,
/* Especifica o Execution Mode */
-- INDEX chave_estrangeira3 (ExecutionMode_Cod) ,
FOREIGN KEY (ExecutionMode_Cod) REFERENCES ExecutionMode(Cod)
ON DELETE NO ACTION);

--  ------ BLOCKATIVITY --------------------------------

create table BlockActivity(
Package_Id varchar(18) NOT NULL,
Activity_Id varchar(18) NOT NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
ActivityType_Cod integer NOT NULL,
ActivitySet_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Activity_Id),
/* Especializacao de Activity*/
-- INDEX chave_estrangeira1 (Package_Id,WorkflowProcess_Id,Activity_Id,ActivityType_Cod),
FOREIGN KEY (Package_Id,WorkflowProcess_Id,Activity_Id,ActivityType_Cod) REFERENCES
Activity(Package_Id,WorkflowProcess_Id,Id,ActivityType_Cod)
ON DELETE CASCADE,
/* Fixa que o valor de ActivityType_Cod deve ser igual 3 */
-- INDEX chave_estrangeira2 (ActivityType_Cod),
FOREIGN KEY (ActivityType_Cod) REFERENCES ActivityType_BlockActivity(Cod)
ON DELETE NO ACTION,
/* Relacao com ActivitySet */
```

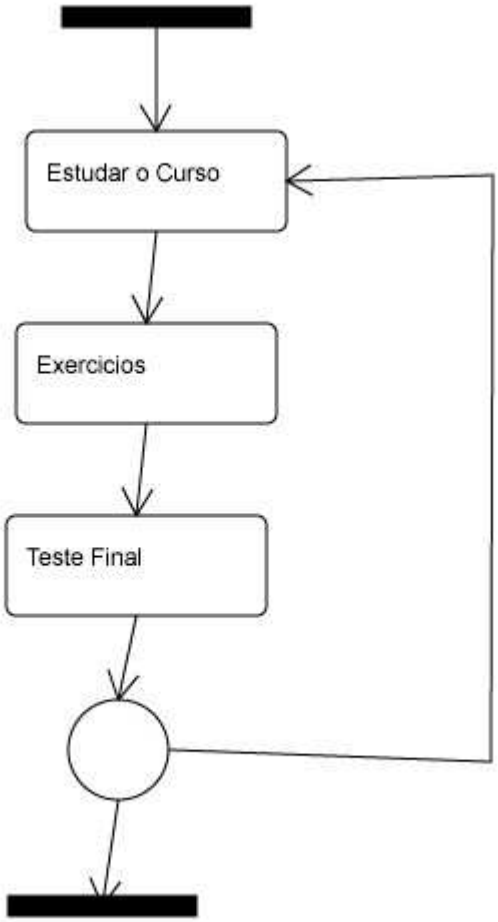
```
-- INDEX chave_estrangeira3 (Package_Id,WorkflowProcess_Id,ActivitySet_Id),
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id) REFERENCES
ActivitySet(Package_Id,WorkflowProcess_Id,Id)
ON DELETE NO ACTION);

create table ActualParameter_Subflow(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Parameter varchar(25) NOT NULL,
Activity_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod),

-- INDEX (Package_Id,Activity_Id),
FOREIGN KEY (Package_Id,Activity_Id) REFERENCES Subflow(Package_Id,Activity_Id)
ON DELETE CASCADE);

create table ActualParameter_Tool(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Parameter varchar(25) NOT NULL,
Activity_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod),

-- INDEX (Package_Id,Activity_Id),
FOREIGN KEY (Package_Id,Activity_Id) REFERENCES Tool(Package_Id,Activity_Id)
ON DELETE CASCADE);

create table ExtendedAttribute_Activity(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Activity */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES Activity(Package_Id,Id)
ON DELETE CASCADE);

create table ExtendedAttribute_Tool(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Tool */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES Tool(Package_Id,Activity_Id)
ON DELETE CASCADE);

create table Join_Split_Type(
Cod integer PRIMARY KEY,
Type varchar(5));

/* Enumeracao. Tabela Fixa */
insert into Join_Split_Type (Cod,Type) values (1,'AND');
insert into Join_Split_Type (Cod,Type) values (2,'XOR');

create table TransitionRestriction(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Activity_Id varchar(18) NOT NULL,
ActivitySet_Id varchar(18) NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
JoinType integer NULL,
SplitType integer NULL,
PRIMARY KEY (Package_Id,Cod),
```

```
UNIQUE (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod),
UNIQUE (Package_Id,WorkflowProcess_Id,Cod),
/* Quando a Activity pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Activity_Id) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Activity_Id) REFERENCES Activity
(Package_Id,WorkflowProcess_Id,ActivitySet_Id,Id)
ON DELETE NO ACTION,
/* Quando a Activity NAO pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,Activity_Id) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,Activity_Id) REFERENCES Activity (Package_Id,WorkflowProcess_Id,Id)
ON DELETE CASCADE,
/* Especifica o tipo do Join */
-- INDEX (JoinType),
FOREIGN KEY (JoinType) REFERENCES Join_Split_Type (Cod)
ON DELETE NO ACTION,
/* Especifica o tipo do Split */
-- INDEX (SplitType),
FOREIGN KEY (SplitType) REFERENCES Join_Split_Type (Cod)
ON DELETE NO ACTION);

create table Split(
Package_Id varchar(18) NOT NULL,
ActivitySet_Id varchar(18) NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
SplitType integer NOT NULL,
PRIMARY KEY (Package_Id,Cod),
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod),
-- INDEX (Package_Id,WorkflowProcess_Id,Cod),
/* Quando a Activity pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod) REFERENCES TransitionRestriction
(Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod)
ON DELETE NO ACTION,
/* Quando a Activity NAO pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,Cod) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,Cod) REFERENCES TransitionRestriction
(Package_Id,WorkflowProcess_Id,Cod)
ON DELETE CASCADE,
/* Especifica o tipo do Split */
-- INDEX (SplitType),
FOREIGN KEY (SplitType) REFERENCES Join_Split_Type (Cod)
ON DELETE NO ACTION);

create table ConditionType(
Cod integer PRIMARY KEY,
Type varchar(25));

/* Tabela de enumeracao. Fixa */
insert into ConditionType (Cod,Type) values (1,'CONDITION');
insert into ConditionType (Cod,Type) values (2,'OTHERWISE');
insert into ConditionType (Cod,Type) values (3,'EXCEPTION');
insert into ConditionType (Cod,Type) values (4,'DEFAULTEXCEPTION');

create table Transition(
Package_Id varchar(18) NOT NULL,
Id varchar(18) NOT NULL,
ActivitySet_Id varchar(18) NULL,
WorkflowProcess_Id varchar(18) NOT NULL,
Name varchar(25) NULL,
Description varchar(25) NULL,
FromActivity varchar(18) NOT NULL,
ToActivity varchar(18) NOT NULL,
Condition_Cod integer,
ConditionType_Cod integer NULL,
TransitionRestriction_Cod integer NULL,
PRIMARY KEY (Package_Id,Id),
FOREIGN KEY (Package_Id) REFERENCES Package(Id)
ON DELETE CASCADE,
/* Relacao FROM. Quando a Transicao pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,FromActivity) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id,FromActivity) REFERENCES Activity
(Package_Id,WorkflowProcess_Id,ActivitySet_Id,Id)
ON DELETE NO ACTION,
/* Relacao TO. Quando a Transicao pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,ToActivity) ,
```

```
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id,ToActivity) REFERENCES Activity
(Package_Id,WorkflowProcess_Id,ActivitySet_Id,Id)
ON DELETE NO ACTION,
/* Relacao From. Quando a Transicao NAO pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,FromActivity) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,FromActivity) REFERENCES Activity (Package_Id,WorkflowProcess_Id,Id)
ON DELETE NO ACTION,
/* Relacao TO. Quando a Transicao NAO pertence a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ToActivity) ,
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ToActivity) REFERENCES Activity (Package_Id,WorkflowProcess_Id,Id)
ON DELETE NO ACTION,
/* Relacao de condicao */
-- INDEX (Package_Id,Condition_Cod),
FOREIGN KEY (Package_Id,Condition_Cod) REFERENCES Element (Package_Id,Cod)
ON DELETE NO ACTION,
/* Relacao de TIPO DE condicao */
-- INDEX (ConditionType_Cod),
FOREIGN KEY (ConditionType_Cod) REFERENCES ConditionType(Cod)
ON DELETE NO ACTION,
/* Se pertencer a um Split de um TransitionRestriction que pertenca a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,ActivitySet_Id,TransitionRestriction_Cod),
FOREIGN KEY (Package_Id,WorkflowProcess_Id,ActivitySet_Id,TransitionRestriction_Cod) REFERENCES
TransitionRestriction (Package_Id,WorkflowProcess_Id,ActivitySet_Id,Cod)
ON DELETE NO ACTION,
/* Se pertencer a um Split de um TransitionRestriction que NAO pertenca a um ActivitySet */
-- INDEX (Package_Id,WorkflowProcess_Id,TransitionRestriction_Cod),
FOREIGN KEY (Package_Id,WorkflowProcess_Id,TransitionRestriction_Cod) REFERENCES TransitionRestriction
(Package_Id,WorkflowProcess_Id,Cod)
ON DELETE NO ACTION);

create table ExtendedAttribute_Transition(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Transition */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES Transition(Package_Id,Id)
ON DELETE CASCADE);

create table ExtendedAttribute_Participant(
Package_Id varchar(18) NOT NULL,
Cod integer NOT NULL,
Name varchar(25) NOT NULL,
Value varchar(25),
Parent_Id varchar(18) NOT NULL,
PRIMARY KEY (Package_Id,Cod) ,
/* Eh uma especializacao de Element */
-- INDEX chave_estrangeira2 (Package_Id,Cod),
FOREIGN KEY (Package_Id,Cod) REFERENCES Element(Package_Id,Cod)
ON DELETE NO ACTION,
/* Relaciona-se com Participant */
-- INDEX chave_estrangeira3 (Package_Id,Parent_Id),
FOREIGN KEY (Package_Id,Parent_Id) REFERENCES Participant(Package_Id,Id)
ON DELETE CASCADE);

/*tabelas da maquina de workflow*/
CREATE TABLE [dbo].[EntMaq_Conferencia] ([Id] [int] IDENTITY (1, 1) NOT NULL ,
[Titulo] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
[Responsavel] [int] NOT NULL ,[Data_Inicio] [datetime] NULL ,
[Data_Fim] [datetime] NULL ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Conferencia_Participante] ([Conferencia_Id] [int] NULL ,
[Participante_Id] [int] NULL ,[Data_Inicio] [datetime] NULL ,
[Data_Fim] [datetime] NULL ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Cooperacao] ([Id_Reg] [int] IDENTITY (1, 1) NOT NULL ,
[Package_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NOT NULL ,
```

```
[Id] [varchar] (18) COLLATE Latin1_General_CI_AS NOT NULL ,
[Responsavel_Id] [int] NOT NULL ,[Editando] [bit] NOT NULL ,
[Ativo] [bit] NULL ,[Data_Inicial] [datetime] NULL ,
[Data_Final] [datetime] NULL ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Historico] ([Id] [int] IDENTITY (1, 1) NOT NULL ,
[Package_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[WorkflowProcess_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[Application_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[ActivitySet_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[Activity_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[Participant_Id] [varchar] (18) COLLATE Latin1_General_CI_AS NULL ,
[Usuario_Id] [int] NULL ,[Data_Ocorrencia] [datetime] NULL ,
[Tipo] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Modulo] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Comando] [varchar] (500) COLLATE Latin1_General_CI_AS NULL ,
[Observacoes] [varchar] (500) COLLATE Latin1_General_CI_AS NULL ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Mensagens] (       [Id] [int] IDENTITY (1, 1) NOT NULL ,
[De_Usuario] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Para_Usuario] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Titulo] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Mensagem] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Data_Envio] [datetime] NULL ,[Modo_Email] [bit] NULL ,
[Modo_Interface] [bit] NULL ) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Parametros] ([Período_Antecedencia] [datetime] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[EntMaq_Usuarios] ([Id] [int] IDENTITY (1, 1) NOT NULL ,
[Nome] [varchar] (50) COLLATE Latin1_General_CI_AS NOT NULL ,
[Login] [varchar] (10) COLLATE Latin1_General_CI_AS NOT NULL ,
[Senha] [varchar] (10) COLLATE Latin1_General_CI_AS NOT NULL ,
[email] [varchar] (100) COLLATE Latin1_General_CI_AS NULL ,
[Cidade] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Estado] [varchar] (20) COLLATE Latin1_General_CI_AS NULL ,
[Pais] [varchar] (20) COLLATE Latin1_General_CI_AS NULL ,
[Fone] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Formacao] [varchar] (50) COLLATE Latin1_General_CI_AS NULL ,
[Obs] [varchar] (300) COLLATE Latin1_General_CI_AS NULL ,
[DataNascimento] [datetime] NULL ) ON [PRIMARY]
GO
```

Anexo 2 Definição Gráfica do Workflow



Anexo 3 Definição em XPDL do Workflow

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://www.wfmc.org/2002/xpdl1.0" Id="Package1"
Name="WorkflowPackage">
     <PackageHeader>
          <XPDLVersion>0.09</XPDLVersion>
          <Vendor>Luis Mota</Vendor>
          <Created>Saturday 25 October 2003
          </Created>
          <Description>EnsinoOnLine</Description>
     </PackageHeader>
     <Participants>
          <Participant id="SYSTEM">
               <ParticipantType Type="SYSTEM"/>
                    <Description> Automatic Participant
                         (default)
                    </Description>
          </Participant>
          <Participant id="ROLE">
               <ParticipantType Type="ROLE"/>
               <Description>Human Participant</Description>
          </Participant>
     </Participants>
     <Applications>
          <Application Id="Casati Comands">
             <FormalParameters>
                    <FormalParameter Id="Comands" Index="1" Mode="IN">
                         <DataType>
                              <BasicType Type="STRING"/>
                         </DataType>
                    </FormalParameter>
               </FormalParameters>
          </Application>
          <Application Id="Casati Exception">
               <FormalParameters>
                    <FormalParameter Id="Exception" Index="1"
                         Mode="IN">
                         <DataType>
                              <BasicType Type="STRING"/>
                         </DataType>
                    </FormalParameter>
               </FormalParameters>
          </Application>
          <Application Id="Casati Begin">
               <FormalParameters>
                    <FormalParameter Id="Begin" Index="1" Mode="IN">
                         <DataType>
                              <BasicType Type="STRING"/>
                         </DataType>
                    </FormalParameter>
               </FormalParameters>
          </Application>
          <Application Id="Casati End">
               <FormalParameters>
                    <FormalParameter Id="End" Index="1" Mode="IN">
                         <DataType>
                              <BasicType Type="STRING"/>
                         </DataType>
```

```
                </FormalParameter>
            </FormalParameters>
        </Application>
        <Application Id="Dispatcher">
            <FormalParameters>
                <FormalParameter Id="Dispatcher" Index="1"
                    Mode="IN">
                    <DataType>
                        <BasicType Type="STRING"/>
                    </DataType>
                </FormalParameter>
            </FormalParameters>
        </Application>
        <Application Id="Counter">
            <FormalParameters/>
        </Application>
    </Applications>
    <WorkflowProcesses>
    <WorkflowProcess Id="luis" Name="luis" AccessLevel="Public">
        <ProcessHeader/>
        <Activities>
        <Activity Id="element" name="Begin">
            <Implementation>
                <Tool Id="Casati Begin" Type="APPLICATION">
                    <ActualParameters>
                        <ActualParameter>Begin</ActualParameter>
                    </ActualParameters>
                </Tool>
            </Implementation>
        </Activity>
        <Activity Name= "Estudar o Curso" Id= "element1" >
            <Implementation>
                <No/>
            </Implementation>
            <Performer>SYSTEM</Performer>
        </Activity>
        <Activity Name= "Exercicios" Id= "element2" >
            <Implementation>
                <No/>
            </Implementation>
            <Performer>SYSTEM</Performer>
        </Activity>
        <Activity Name= "Teste Final" Id= "element3" >
            <Implementation>
                <No/>
            </Implementation>
            <Performer>SYSTEM</Performer>
         </Activity>
         <Activity Id="element4">
            <Route/>
            <TransitionRestrictions>
                <TransitionRestriction>
                    <Split Type="AND">
                        <TransitionRefs>
                        <TransitionRef Id="connector4"/>
                        <TransitionRef Id="connector5"/>
                        </TransitionRefs>
                    </Split>
                </TransitionRestriction>
            </TransitionRestrictions>
        </Activity>
```

```
          <Activity Id="element5" name="">
               <Implementation>
                    <Tool Id="Casati End" Type="APPLICATION">
                         <ActualParameters>
                              <ActualParameter>End</ActualParameter>
                         </ActualParameters>
                    </Tool>
               </Implementation>
          </Activity>
          </Activities>
          <Transitions>
          <Transition Id="connector" From="element" To="element1"/>
          <Transition Id="connector1" From="element1" To="element2"/>
          <Transition Id="connector2" From="element2" To="element3"/>
          <Transition Id="connector3" From="element3" To="element4"/>
          <Transition Id="connector4" From="element4" To="element5">
               <condition>"Nota >= 7"</condition>
          </Transition>
          <Transition Id="connector5" From="element4" To="element1">
               <condition>"Nota < 7"</condition>
          </Transition>
          </Transitions>
     </WorkflowProcess>
     </WorkflowProcesses>
</Package>
```

Anexo 4 Script em SQL para inserção da Definição do Workflow

INSERT INTO Package (Id, Name, XPDLVersion, Vendor, Created, Description, Documentation, PriorityUnit, CostUnit, PublicationStatus, Author, Version,
Codepage, CountryKey, GraphConformance) VALUES ('PackageCurso', 'WorkflowPackage', '0.09', 'Luis Mota',
'25 October 2003', 'Ensino', NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, 3)

INSERT INTO Participant (Package_Id, Id, WorkflowProcess_Id, Name, ParticipantType_Cod, Description, Xref, Location, Namespace) VALUES ('PackageCurso', 'SYSTEM', NULL, NULL, 6, 'Automatic Participant (default)', NULL, NULL, NULL)

INSERT INTO Participant (Package_Id, Id, WorkflowProcess_Id, Name, ParticipantType_Cod, Description, Xref, Location, Namespace) VALUES ('PackageCurso', 'ROLE', NULL, NULL, 3, 'Human Participant', NULL, NULL, NULL)

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Casati Comands', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Casati Comands', 2)

INSERT INTO DataType (Package_Id, Cod, Type) VALUES ('PackageCurso', 2, 1)

INSERT INTO BasicType (Package_Id, Cod, DataType_Type, Type) VALUES ('PackageCurso', 2, 1, 1)

INSERT INTO FormalParameter_Application (Package_Id, Id, Indice, FormalParameter_Mode_Cod, DataType_Cod, Application_Id) VALUES ('PackageCurso', 'Comands', '1', 1, 2, 'Casati Comands')

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Casati Exception', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Casati Exception', 2)

INSERT INTO DataType (Package_Id, Cod, Type) VALUES ('PackageCurso', 3, 1)

INSERT INTO BasicType (Package_Id, Cod, DataType_Type, Type) VALUES ('PackageCurso', 3, 1, 1)

INSERT INTO FormalParameter_Application (Package_Id, Id, Indice, FormalParameter_Mode_Cod, DataType_Cod, Application_Id) VALUES ('PackageCurso', 'EXCEPTION', '1', 1, 3, 'Casati Exception')

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Casati Begin', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Casati Begin', 2)

INSERT INTO DataType (Package_Id, Cod, Type) VALUES ('PackageCurso', 4, 1)

INSERT INTO BasicType (Package_Id, Cod, DataType_Type, Type) VALUES ('PackageCurso', 4, 1, 1)

INSERT INTO FormalParameter_Application (Package_Id, Id, Indice, FormalParameter_Mode_Cod, DataType_Cod, Application_Id) VALUES ('PackageCurso', 'Begin', '1', 1, 4, 'Casati Begin')

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Casati End', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Casati End', 2)

INSERT INTO DataType (Package_Id, Cod, Type) VALUES ('PackageCurso', 5, 1)

INSERT INTO BasicType (Package_Id, Cod, DataType_Type, Type) VALUES ('PackageCurso', 5, 1, 1)

INSERT INTO FormalParameter_Application (Package_Id, Id, Indice, FormalParameter_Mode_Cod, DataType_Cod, Application_Id) VALUES ('PackageCurso', 'End', '1', 1, 5, 'Casati End')

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Dispatcher', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Dispatcher', 2)

INSERT INTO DataType (Package_Id, Cod, Type) VALUES ('PackageCurso', 6, 1)

INSERT INTO BasicType (Package_Id, Cod, DataType_Type, Type) VALUES ('PackageCurso', 6, 1, 1)

INSERT INTO FormalParameter_Application (Package_Id, Id, Indice, FormalParameter_Mode_Cod, DataType_Cod, Application_Id) VALUES ('PackageCurso', 'Dispatcher', '1', 1, 6, 'Dispatcher')

INSERT INTO Application (Package_Id, Id, WorkflowProcess_Id, ApplicationChoice_Cod, Name, Description) VALUES ('PackageCurso', 'Counter', NULL, 2, NULL, NULL)

INSERT INTO Application_FormalParameter (Package_Id, Id, ApplicationChoice_Cod) VALUES ('PackageCurso', 'Counter', 2)

INSERT INTO WorkflowProcess (Package_Id, Id, Name, AccessLevel_Cod) VALUES ('PackageCurso', 'luis', 'luis', 1)

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element', 'luis', 1, 'Begin', NULL)

UPDATE WorkflowProcess SET FirstActivity = '1' WHERE Package_Id = 'PackageCurso' AND Id = 'luis'

UPDATE Activity SET ActivityType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Id = 'element'

INSERT INTO Implementation (ActivityType_Cod, ImplementationType_Cod, Package_Id, Activity_Id) VALUES (2, 2, 'PackageCurso', 'element')

UPDATE Implementation SET ImplementationType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Activity_Id = 'element'

INSERT INTO Tool (ToolType_Cod, ImplementationType_Cod, Package_Id, Activity_Id, Id, Description) VALUES (1, 2, 'PackageCurso', 'element', 'Casati Begin', NULL)

INSERT INTO ActualParameter_Tool (Package_Id, Cod, Parameter, Activity_Id) VALUES ('PackageCurso', 7, 'Begin', 'element')

UPDATE Tool SET Description = NULL WHERE Package_Id = 'PackageCurso' AND Activity_Id = 'element'

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = NULL, Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element'

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element1', 'luis', 1, 'Estudar o Curso', NULL)

UPDATE Activity SET ActivityType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Id = 'element1'

INSERT INTO Implementation (ActivityType_Cod, ImplementationType_Cod, Package_Id, Activity_Id) VALUES (2, 1, 'PackageCurso', 'element1')

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = 'SYSTEM', Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element1'

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element2', 'luis', 1, 'Exercicios', NULL)

UPDATE Activity SET ActivityType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Id = 'element2'

INSERT INTO Implementation (ActivityType_Cod, ImplementationType_Cod, Package_Id, Activity_Id) VALUES (2, 1, 'PackageCurso', 'element2')

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = 'SYSTEM', Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element2'

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element3', 'luis', 1, 'Teste Final', NULL)

UPDATE Activity SET ActivityType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Id = 'element3'

INSERT INTO Implementation (ActivityType_Cod, ImplementationType_Cod, Package_Id, Activity_Id) VALUES (2, 1, 'PackageCurso', 'element3')

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = 'SYSTEM', Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element3'

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element4', 'luis', 1, NULL, NULL)

INSERT INTO TransitionRestriction (Package_Id, Cod, Activity_Id, ActivitySet_Id, WorkflowProcess_Id) VALUES ('PackageCurso', 8, 'element4', NULL, 'luis')

UPDATE TransitionRestriction SET JoinType = NULL, SplitType = 1 WHERE Package_Id = 'PackageCurso' AND Cod = 8

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = NULL, Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element4'

INSERT INTO Activity (Package_Id, Id, WorkflowProcess_Id, ActivityType_Cod, Name, ActivitySet_Id) VALUES ('PackageCurso', 'element5', 'luis', 1, '', NULL)

UPDATE Activity SET ActivityType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Id = 'element5'

INSERT INTO Implementation (ActivityType_Cod, ImplementationType_Cod, Package_Id, Activity_Id) VALUES (2, 2, 'PackageCurso', 'element5')

UPDATE Implementation SET ImplementationType_Cod = 2 WHERE Package_Id = 'PackageCurso' AND Activity_Id = 'element5'

INSERT INTO Tool (ToolType_Cod, ImplementationType_Cod, Package_Id, Activity_Id, Id, Description) VALUES (1, 2, 'PackageCurso', 'element5', 'Casati End', NULL)

INSERT INTO ActualParameter_Tool (Package_Id, Cod, Parameter, Activity_Id) VALUES ('PackageCurso', 9, 'End', 'element5')

UPDATE Tool SET Description = NULL WHERE Package_Id = 'PackageCurso' AND Activity_Id = 'element5'

UPDATE Activity SET Description = NULL, StartMode = NULL, FinishMode = NULL, Limite = NULL, Performer = NULL, Priority = NULL, Icon = NULL, Documentation = NULL, Cost = NULL, WaitingTime = NULL, WorkingTime = NULL, Duration = NULL, Instantiation_Cod = NULL WHERE Package_Id = 'PackageCurso' AND Id = 'element5'

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector', NULL, NULL, 'element', 'element1', NULL, NULL)

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector1', NULL, NULL, 'element1', 'element2', NULL, NULL)

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector2', NULL, NULL, 'element2', 'element3', NULL, NULL)

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector3', NULL, NULL, 'element3', 'element4', NULL, NULL)

INSERT INTO Element (Package_Id, Cod, Name, Parent) VALUES ('PackageCurso', 10, 'Condition', NULL)

UPDATE Element SET Content = 'Nota >= 7', Name = NULL WHERE Package_Id = 'PackageCurso' AND Cod = 10

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector4', NULL, NULL, 'element4', 'element5', 10, NULL)

INSERT INTO Element (Package_Id, Cod, Name, Parent) VALUES ('PackageCurso', 11, 'Condition', NULL)

UPDATE Element SET Content = 'Nota < 7', Name = NULL WHERE Package_Id = 'PackageCurso' AND Cod = 11

INSERT INTO Transition (Package_Id, WorkflowProcess_Id, ActivitySet_Id, Id, Name, Description, FromActivity, ToActivity, Condition_Cod, ConditionType_Cod) VALUES ('PackageCurso', 'luis', NULL, 'connector5', NULL, NULL, 'element4', 'element1', 11, NULL)

Referências

[AMA 97] AMARAL, V. **Técnicas de Modelagem de Workflow.** 1997. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[APP 2002] APPLETON, B. **Patterns and Softwar**e: Essential Concepts and Terminology. Disponível em: <http://www.enteract.com/~bradapp /docs/patterns-intro.html>. Acesso em: nov.2002.

[BAI 2001] BAINA, K. et al. **CORVETTE:** building a cooperative workflow system by plugging together a workflow management system and a cooperative trasaction manager (experiment report). Disponível em: <http://www.loria.fr/equipes/ecoo/corvette/papers/experimentReport/ experimentReport.pdf>. Acesso em: jun.2003.

[BRI 97] BRINKS, T; MCDANIEL, S. Small group discussions. In: WORKSHOP ON AWARENESS IN COLLABORATIVE SYSTEMS, CHI, 1997. **Proceedings…** Disponível em: <http://www.usabilityfirst.com/groupware/awareness/workshop/small-groups.html>. Acesso em: nov.2002.

[BRE 2001] BRITTO, E. C. de S. **Um Estudo de Critérios da Qualidade para Avaliação de Modelos de *Workflow***. 2001. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[BYZ 2002] OPERA GROUP. **Byzance**. Disponível em: <http://www.inrialpes.fr/opera/Byzance.fr.html>. Acesso em: nov.2002.

[CEP 2001] CEPEDA, A. R. **Validação de um *Workflow* de Autoria na Implementação de um Curso de Ensino à Distância** . 2001. (Curso de Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[CLA 2002] CLAVELEIRA, C. **Les Applications de Travail Collaborati**f. Disponível em: <http://www.cru.fr/multimedia/applis_multi/applis_multi.fm.html>. Acesso em: nov.2002.

[CSM 2003] Core Standards For Markup Language Technologies. Disponível em: <http://xml.coverpages.org/coreStandards.html>. Acesso em: jan.2003.

[CSS 2003] W3C Cascading Style Sheets. Disponível em: <http://xml.coverpages.org/css.html>. Acesso em: jan.2003.

[DAV 2002] DAVIS, T. E. Clever Facade makes JDBC look easy. Cool Tools-**JavaWorl**d, May, 1999. Disponível em: <http://www.javaworld.com/jw-05-1999/jw-05-cooltools.html>. Acesso em: nov.2002.

[DIE 96] DIETRICH, E. **Projeto de um sistema de Suporte à Autoria Cooperativa de Hiperdocumento**s. 1996. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[EDE 94] EDELWEISS, N. **Sistemas de Informação de Escritórios: Um Modelo para Especificações Temporais**. 1994. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[EXT 2003] EXTENSIBLE Stylesheet Language (XSL). Disponível em: <http://xml.coverpages.org/xsl.html>. Acesso em: jan.2003.

[FIS 2001] FISCHER, L. ***Workflow Handbook***. Florida: Future Strategies Inc., Book Division, 2001.

[FRA 2003] FRANÇA, MONTGOMERY B. **Um Compilador para XPDL**. 2003. Semana Acadêmica (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[GMD 2002] GMD. **Basic Support for Cooperative Wor**k. Disponível em: <http://orgwis.gmd.de/projects/BSCW>. Acesso em: nov.2002.

[GMD 2002a] GMD. **BSC**W. Disponível em: <http://bscw.gmd.de>. Acesso em: nov.2002.

[GRE 98] GREENBERG, Saul; ROSEMAN, Mark. Groupware toolkits for synchronous work. In: BEAUDOUIN-LAFON, M. (Ed.**). Computer-Supported Cooperative Work:** trends in software series. [S.l.]: John Wiley & Sons, 1998. p. 135-168. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: nov.2002.

[GRE 2002] GREENBERG, S. **GroupLab:** Laboratory for Computer Supported Cooperative Work & Human Computer Interaction. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab>. Acesso em: nov.2002.

[GRI 2000] GRIGORI, D.; Hala Skaf-Molli and François Charoy. **Adding Flexibility in a Cooperative Workflow Execution Engine**. Disponível em: <http://www.loria.fr/~skaf/paper/hcpn00/FlexLncs.htm>. Acesso em: jun.2003.

[GOD 99] GODART, C.; Perrin, O.; Skaf, H. **COO:** a workflow operator to improve cooperation modeling in virtual processes. Disponível em: <http://citeseer.nj.nec.com/godart99coo.html>. Acesso em: jun.2003.

[GOD 2001] GODART, C.; Grigori, D.; Akifuji, S. **A component based approach for the development of workflow systems to support cooperative and competitive process**. Disponível em: <http://www.loria.fr/equipes/ecoo/corvette/papers/synthesis/synthesis pdf>. Acesso em: jun.2003.

[GOD 2000] GODART, C. et al. **Cooperative Workflows to Coordinate Asynchronous Cooperative Applications in a Simple Way**. Disponível em: <http://www.loria.fr/~skaf/paper/PADS00/pads00.pdf>. Acesso em: jun.2003.

[INR 2002] INRIA – PROJETO OPERA. ***Madeus – Un éditeur de documents multimédia***. Disponível em: <http://www.inrialpes.fr/opera/Madeus.fr.html>. Acesso em: jan.2002.

[INR 2002a] INRIA – PROJETO OPERA. ***RR-2983 - Presentation Services in MADEUS:*** *an Authoring Environment for Multimedia Documents*. Disponível em: <http://www.inria.fr/rrrt/rr-2983.html>. Acesso em: jan.2002.

[INT 2002] INTERNET Engineering Task Force. Disponível em: <http://www.ietf.org>. Acesso em: nov.2002.

[IOC 2002] IOCHPE, C.; THOM, L. H. ***Inferring Aspects Of The Organizational Structure Through Workflow Process Analysis***. In: International Conference on Enterprise Information Systems, ICEIS 4., 2002. **Proceedings...** Setubal: Escola Superior de Tecnologia de Setúbal, 2002. v2, p. 758 – 763.

[JUN 2000] JUNGBLUT, M. V. **Modelagem de *Workflow* :** Estudo de Caso Relacionado ao Tribunal Regional do Trabalho da 4º Região e Comparação do Modelo de Casati/Ceri e Modelo TF-ORM. 2000. (Curso de Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[KIR 99] KIRSCH PINHEIRO, M. **Edição Cooperativa de Hiperdocumentos na WWW .** 1999. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[KIR 2001] KIRSCH PINHEIRO, M. **Mecanismo de Suporte à Percepção em Ambientes Cooperativos**. 2001. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[KIR 2002] KIRSCH PINHEIRO, M.; LIMA, J.V.; BORGES, M.R. S. **A Framework for Awareness in CSCW System**s. Disponível em: <http://www.inf.ufrgs.br/~manuele/ArtigoAwareness-25pag.doc.zip>. Acesso em: nov.2002.

[KIR 2002a] KIRSCH PINHEIRO, M.; LIMA, J.V.; BORGES, M.R. S. **Awareness in Groupware System**s. Disponível em: <http://www.inf.ufrgs.br/~manuele/awareness.zip>. Acesso em: nov.2002.

[KUN 2001] KUNDE, G. F.; SOUTO, M. A.; PALAZZO, J. M. de O. **Evolução Dinâmica de um Curso a Distância Modelado por *Workflow***. Porto Alegre: Revista Brasileira de Informática na Educação Vol 9, Set-2001. Pág 35/50.

[LAB 98] LABIDI, S.; HAMMOUDI, S. **Collaborative Workflow Management in WEICOT**. Disponível em: <http://citeseer.ist.psu.edu/labidi98collaborative.html>. Acesso em: jun.2003.

[LAM 2003] LAMBRECHTS, F. **Aprendendo XML.** Disponível em: <http://www.linuxfocus.org/Portugues/May2002/article242.shtml>. Acesso em: jan.2003.

[LEI 99] LEITE, L. L.; LIMA, J. V. de. Modelos de tempo para apresentações multimídia complexas. In: SEMANA ACADÊMICA DO PPGC/UFRGS, 4., 1999, Porto Alegre, RS. **Anais…** Porto Alegre: PPGC/UFRGS, 1999. Disponível em: <http://www.inf.ufrgs.br/pos/SemanaAcademica/Semana99/leticia/leticia.html>. Acesso em: nov.2002.

[MAC 2003] MACORATTI. **XML – Introdução**. Disponível em: <http://www.macoratti.net/xml.htm>. Acesso em: jan.2003.

[NIC 98] NICOLAO, MARIANO. **Análise da adequação do Modelo de *Workflow* para Bases de Conhecimento em Sistemas de Tutores Inteligentes**. 1998. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[NIC 99] NICOLAO, MARIANO. **Modelagem de *Workflow* Utilizando um Modelo de Dados Temporal Orientado a Objetos com Papéis**. 1999. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[OPE 2001] *ÓPERA Research Group*. Disponível por WWW em <http://www.inrialpes.fr/opera>. Acesso em: nov.2001.

[PAR 98] PARSOWITH, S; et al. **TickerTape:** notification and communication in a single line. Disponível em: <http://elvin.dstc.com/doc/papers/apchi98/apchi98.pdf>. Acesso em: nov.2002.

[PRI 2002] PRINZ, W. **The POLITeam Awareness Clien**t. Disponível em: <http://orgwis.gmd.de/projects/POLITeam/poliawac>. Acesso em: nov.2002.

[PRO 2002] PROJECT POLITEAM. **POLITea**m. Disponível em: <http://orgwis.gmd.de/projects/POLITeam>. Acesso em: nov.2002.

[QUI 2002] QUINT, V.; VATTON, I. **Introduction to Amay**a. Disponível em: <http://www.w3c.org/pub/WWW/TR/NOTE-Amaya-970220>. Acesso em: nov.2002.

[ROS 96] ROSEMAN, M.; GREENBERG, S. Building real time groupware with GroupKit, a groupware toolkit. **ACM Transactions on Computer Human Interaction**s, New York, v.1, n.3, p. 66-106, Mar. 1996. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: nov.2002.

[ROS 97] ROSEMAN, M.; GREENBERG, S. Building groupware with GroupKit. In: HARRISON, M. (Ed.). **Tcl/Tk Tool.** [S.l.]: O'Reilly Press, 1997.p.535-564. Disponível em: <http://www.cpsc.ucalgary.ca/grouplab/papers>. Acesso em: nov.2002.

[SCH 98] SCHMIDT, A; et al. An Agent-Based Telecooperation Framework. COBUILD, 1998. **Proceedings** Darmstadt, 1998. Disponível em: <http://www-vs.informatik.uni-ulm.de/Papers/cobuild98/cobuild98.os >. Acesso em: nov.2002.

[SIZ 99] SIZILIO, G. R. M. A. **Estudo de Técnicas de Modelagem de *Workflow* Aplicada a Modelagem de Cursos de Educação á Distância**. 1999. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[SOH 98] SOHLENKAMP, M. **Supporting group awareness in multi-user enviroment through perceptualizatio**n. Paderborn: Fachbereich Mathematik-Informatik der Universität - Gesamthochschule, 1998. Disponível em: <http://orgwis.gmd.de/projects/POLITeam/poliawac/ms-diss>. Acesso em: nov.2002.

[SOU 96] SOUZA, A. S. de. **Um Estudo Sobre Trabalho Cooperativo Suportado Por Computador (CSCW).** 1996. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[STA 2003] STANDARD Generalized Markup Language (SGML). Disponível em: <http://xml.coverpages.org/sgml.html>. Acesso em: mar.2003.

[TEL 2001] TELECKEN, T.; et al. **Introdução de Percepção de Eventos Passados no *Software Byzance***. Projeto CEMT - Relatório, Ago - 2001. Disponível em: <http://gnome.inrialpes.fr:1959/PT/BW-Byzance_rapportPT.html>. Acesso em: mar.2003.

[TEO 2003] TEODOSIO, F. **O que é XML?** Disponível em: <http://www.sociteo.com/fernandoteodosio/ficheiros/xml.htm>. Acesso em: jan.2003.

[THO 2001] *THOT - Structured Document Editor*. Disponível por WWW em <http://www.inrialpes.fr/opera/Thot.en.html>. Acesso em: out.2001

[W3C 2003] W3C Scalable Vector Graphics (SVG). Disponível em: <http://xml.coverpages.org/svg.html>. Acesso em: jan.2003.

[WIL 2003] WILLIGHAGEN, E. **Usando o XML e o XSLT para contruir o LinuxFocus.org(/Nederlands).** Disponível em: <http://www.linuxfocus.org/Portugues/September2001/article206.shtml>. Acesso em: jan.2003.

[WOR 96] WORKFLOW MANAGEMENT COALITION. ***Terminology & Glossary***. Document Number WfMC-TC-1011, Document Status – Issue 2.0, June 1996. Disponível em: <http://kyonggi.ac.kr/~jinun/research/workflow/wfmc/glossary.pdf>. Acesso em: jan.2003.

[WOR 99] WORKFLOW MANAGEMENT COALITION. ***Interface 1: Process Definition Interchange Process Model***. Document Number WfMC-TC-1016-P, Document Status – Version 1.1 (Official release), October 1999.

[WOR 2002] WORKFLOW MANAGEMENT COALITION. ***Workflow Process Definition Interface – XML Process Definition Language*** . Document Number WfMC-TC-1025, Document Status – 1.0 Final Draft, October 2002.

[WOR 2003] WORKFLOW MANAGEMENT COALITION. **XML Process Definition Language (XPDL) Version 1.0**. Disponível em: <http://xml.coverpages.org/ni2002-12-10-b.html>. Acesso em: jan.2003.

[WOR 2003a] WORKFLOW MANAGEMENT COALITION. **XML Process Definition Language (XPDL) Beta**. Disponível em: <http://xml.coverpages.org/ni2002-09-10-a.html>. Acesso em: jan.2003.

[XML 2003] XML Linking Language. Disponível em: <http://xml.coverpages.org/xll.html>. Acesso em: jan.2003.

[XML 2003a] XML-Based Process Management Standard. Disponível em: <http://xml.coverpages.org/wfmc19990707.html>. Acesso em: jan.2003.

[XML 2003b] XML and Query Languages. Disponível em: <http://xml.coverpages.org/xmlQuery.html>. Acesso em: jan.2003.

[XML 2003c] XML – Schemas. Disponível em: <http://xml.coverpages.org/schemas.html>. Acesso em: jan.2003.

[XML 2003d] Extensible Markup Language (XML). Disponível em: <http://xml.coverpages.org/xml.html>. Acesso em: jan.2003.

[XML 2003e] XML – Based Workflow and Process Management Standards: XPDL, Wf-XML**.** Disponível em: <http://xml.coverpages.org/wf-xml.html>. Acesso em: jan.2003.

[ZSC 2002] ZSCHORNACK, F. **Estudo sobre Aspectos de Evolução de *Workflow*s**. 2002. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.