Sistemas de Workflow:
Análise da Área e Proposta de Modelo

Paulo Barthelmess

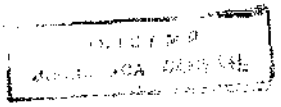
Sistemas de Workflow:
Análise da Área e Proposta de Modelo

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida pelo Sr. **Paulo Barthelmess** e aprovada pela Comissão Julgadora.

Campinas, 25 de junho de 1996.

Prof. Dr. Jacques Wainer _____
Orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do Título de **MESTRE em Ciência da Computação**.



FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Barthelmess, Paulo

B282s Sistemas de Workflow : análise da área e proposta de modelo / Paulo Barthelmess -- Campinas, [S.P. :s.n.], 1996.

Orientador : Jacques Wainer

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Sistemas de informação. 2. Práticas de escritórios - Automação. 3. Modelagem. I. Wainer, Jacques. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Tese defendida e aprovada em 03 de Junho de 1996

Pela Banca Examinadora composta pelos Profs. Drs.

Prof (a). Dr (a). JOSÉ PALAZZO MOREIRA DE OLIVEIRA

Prof (a). Dr (a). CLÁUDIA MARIA BAUZER MEDEIROS

Prof (a). Dr (a). JACQUES WAINER

Agradecimentos

À Ro, pela paciência com o marido estudante e pelo incentivo que nenhuma outra pessoa teria sido capaz de dar.

Ao Prof. Jacques Wainer, que é o autor das melhores idéias que por acaso existam neste trabalho, e que foi uma fonte constante de incentivo, entusiasmo e equilíbrio.

Ao Juliano Oliveira, pelo apoio e influência na área de sua especialidade.

Aos demais colegas que de uma forma ou de outra contribuiram com informações incentivo ou simplesmente boa conversa.

Ao CNPq, pelo apoio financeiro recebido.

Lo! Men have become the tools of their tools.
Henry David Thoreau

Resumo

A presente dissertação enfoca os sistemas de workflow, que se inserem no contexto mais abrangente de software de suporte ao trabalho colaborativo. Sistemas de workflow podem ser definidos como sistemas cujo objetivo é "auxiliar as organizações na especificação, execução, monitoramento e coordenação do fluxo de trabalho em um ambiente de escritório distribuído" [Bul92].

Identificamos, através de análise da literatura da área, fatores estruturais em jogo em sistemas deste tipo, e demonstramos que as abordagens adotadas atualmente cobrem apenas parcialmente o espectro de possibilidades. Identificamos ainda omissões semânticas dos modelos ou especificações. O modelo conceitual que propomos procura tanto ampliar o poder semântico disponível nas direções apontadas pela análise, quanto corrigir os problemas estruturais detectados.

Em especial, são atacados os seguintes problemas:

- Propomos a ampliação do poder semântico através da oferta de um conjunto abrangente de ações básicas e elementos de sincronismo que englobam o tratamento de eventos assíncronos, atividades batch e atividades replicadas;
- Apresentamos os fundamentos para um ambiente de execução fortemente orientado a dados, em que tanto objetos de sistema (como especificações de processo, p.ex.) quanto objetos de aplicação são tratados de maneira uniforme;
- Tratamos o problema da alocação de executores de forma mais abrangente, permitindo a existência de atividades coletivas e o uso de estratégias de alocação diferenciadas, como balanceamento de carga de trabalho, *round-robin* e assim por diante.
- Discutimos também os requisitos adicionais de comunicação introduzidos pela existência de atividades coletivas, cujo objetivo é o de manter a sinergia entre os participantes de cada atividade (a difusão de *awareness*).

Abstract

A special category of collaborative systems, the Workflow Systems, are discussed. Such systems can be defined as "systems that help organizations to specify, execute, monitor, and coordinate the flow of work items within a distributed office environment" [Bul92].

We identify basic structural and semantic issues in such systems, and show that improvements can be made over current systems by offering a better coverage of both aspects.

We then propose a new conceptual model that tries to fill the detected gaps both by providing a stronger, more expressive specification language and a more comprehensive execution environment. In particular, the following issues are covered:

Basic actions and synchronization elements are proposed for asynchronous events, batch and replicated activities.

We propose a strongly data-oriented execution environment, where both application and system objects (such as specifications) area treated in a uniform way.

We present a broader solution to the agent scheduling problem, that allows one to use different allocation strategies, such as load-balancing, round robin and so on, and that lets many agents to be associated to collective activities.

Collective activities give rise to special communication needs, that are treated in the broader awareness diffusion context.

Sumário

1. INTRODUÇÃO

1.1. CONTEXTO

O presente trabalho apresenta uma discussão abrangente sobre sistemas de workflow, suas limitações e oportunidades de melhoria, que incorporamos em um novo modelo conceitual, que pretende solucionar diversos dos problemas estruturais e de modelagem detectados.

Sistemas de workflow se inserem na área mais abrangente de software de apoio ao trabalho cooperativo (*Computer Supported Cooperative Work* - CSCW), cujo objetivo é, como o nome indica, o de propiciar suporte computadorizado a grupos de pessoas que trabalham em conjunto para atingir um objetivo comum. Sistemas de workflow são utilizados em diversas áreas, como a de engenharia de software, a científica e organizacional: nos ateremos no presente trabalho aos voltados para organizações ou escritórios distribuídos.

1.2. DEFINIÇÃO DO PROBLEMA

Sistemas de workflow podem ser definidos como sistemas cujo objetivo é "auxiliar as organizações na especificação, execução, monitoramento e coordenação do fluxo de trabalho em um ambiente de escritório distribuído" [Bul92]. O fluxo de trabalho corresponde à tramitação de documentos e informações entre diversos agentes, pertencentes a unidades organizacionais potencialmente distintas, cada qual agregando uma parcela do trabalho necessário para que se alcance determinado objetivo de negócio.

A observação de que por volta de 90% desta tramitação pode ser considerada rotineira e repetitiva [AGL+93] torna natural que se deseje suprir suporte automatizado para estas tarefas. Após um surto de interesse nos anos 70, associado à área de automação de escritórios, a pesquisa relativa a sistemas de workflow ficou estagnada durante a maior parte dos anos 80, devido ao fracasso dos sistemas pioneiros. Verificou-se que estes sistemas apresentavam uma inflexibilidade que inviabilizava seu uso prático, mesmo em situações simples e controladas.

1.3. LIMITAÇÃO DE SOLUÇÕES EXISTENTES

Na década de 90, ressurge a pesquisa na área, com o aparecimento de sistemas muito mais flexíveis, baseados na experiência de seus predecessores de primeira geração. Apesar de bastante melhores, mesmo estes sistemas apresentam limitações de representação e execução que criam uma oportunidade de melhoria.

Identificamos duas limitações nas soluções existentes:

1) Na maioria dos sistemas propostos, a ênfase se dá quase que exclusivamente em um único aspecto, o da modelagem de processos. Esta ênfase, provavelmente decorrente da abordagem usual da análise de sistemas, cujo foco central é necessariamente o de modelagem, parece não ser diretamente aplicável ao problema da construção de sistemas de workflow;

2) Apesar da ênfase na modelagem, a semântica dos modelos deixa a desejar em muitos aspectos, obrigando os usuários a utilizar artifícios para fazer frente a certas situações de trabalho que identificamos como recorrentes.

1.4. OBJETIVO DO TRABALHO

O objetivo do trabalho e sua principal contribuição está justamente na identificação dos fatores fundamentais em jogo em sistemas deste tipo. Identificamos fatores estruturais, e demonstramos que as abordagens adotadas atualmente cobrem apenas parcialmente o espectro de possibilidades. Identificamos ainda omissões semânticas dos modelos. O modelo conceitual que propomos procura tanto corrigir os problemas estruturais quanto ampliar o poder semântico disponível nas direções apontadas pela análise.

1.5. TÓPICOS

O capítulo 2 define o que são sistemas de workflow e qual o seu interesse como sub-área de pesquisa em sistemas cooperativos. O capítulo 3 procura identificar os fatores de mais alto nível envolvidos na construção de um sistema de workflow, ficando a cargo do capítulo 4 a apresentação dos mecanismos de sincronismo usuais. No capítulo 5 são estabelecidos os requisitos do modelo conceitual (apresentado no capítulo 6). O trabalho encerra com conclusões e trabalhos futuros.

2. O QUE É UM SISTEMA DE WORKFLOW

2.1. INTRODUÇÃO

Apresentaremos neste capítulo uma discussão geral sobre a área de CSCW (*Computer Supported Cooperative Work*), identificando suas peculiaridades, para em seguida examinar os sistemas de workflow de forma mais específica, verificando o que os torna interessantes como tema de pesquisa.

Iniciaremos com uma definição de termos, seguida de discussão sobre sistemas cooperativos em geral, e sistemas de workflow de forma específica. O desafio a ser enfrentado neste tipo de sistema é apresentado em 2.5, sendo complementado por diversos exemplos. O capítulo encerra com conclusões.

2.2. DEFINIÇÃO DE TERMOS

A seguir fornecemos a definição de alguns termos que utilizaremos no decorrer do trabalho.

- **Processo / Workflow** - seqüência de passos necessários para que se possa atingir um determinado objetivo de negócio de uma organização.
- **Especificação / Modelo / Plano** - modelo de um processo com vistas à automação através de um sistema de gerenciamento de workflow (Wfms).
- **Caso** - instância de um workflow, i.é, um caso especifico que se baseia ou segue uma especificação de processo.
- **Agente / Executor** - pessoa ou componente de software capaz de executar uma ou mais tarefas em um caso.
- **Papel** - mecanismo de descrição que representa um determinado agente de acordo com um conjunto pré-estabelecido de habilidades ou conhecimento de contexto necessários à execução de uma tarefa, como por exemplo "gerente", "secretario/a" e assim por diante.
- **Atividade / Tarefa** - uma etapa de um processo, normalmente executada individualmente por um agente.
- **Sistema de workflow / Wfms** - sistema de gerenciamento de workflows (*Workflow Management Systems*). Ambiente de suporte à especificação e execução de processos.
- **Contingência / Exceção** - evento não previsto na especificação de um processo.

2.3. SUPORTE AO TRABALHO COOPERATIVO

Sistemas de workflow se inserem em um contexto mais abrangente, o do groupware, ou software relacionado a CSCW (*Computer Supported Cooperative Work*). Como o nome indica, a área de CSCW compreende todo o software que tem por objetivo prestar auxílio ao trabalho cooperativo. É justamente este o fator que torna este tipo de software diferente dos demais.

A noção de que este tipo de software deve mediar a interação de diversas pessoas que buscam obter um objetivo comum [EGR91] introduz novos requisitos normalmente não encontrados em outros sistemas. Se enfatiza a interação **entre usuários** e não mais a interação sistema/usuário, como acontece na maioria dos sistemas (que passaremos a chamar de convencionais, em contraposição a esta nova classe de software cooperativo ou groupware).

A ubiqüidade de estações individuais de trabalho, ligadas por redes, cria uma oportunidade tecnológica de se prover suporte às atividades de grupos de trabalho nas organizações [AS94]. O padrão de utilização dos computadores migrou de uma centralização representada pelas máquinas de grande porte nos CPDs para um de utilização individual, em que cada usuário ou grupo de usuários possui suas ferramentas, como planilhas e editores, trabalhando normalmente em isolamento. Estas atividades isoladas não correspondem, porém, à real necessidade das organizações, nas quais o trabalho não é realizado costumeiramente por apenas um indivíduo, mas é fruto de um esforço coletivo [Suc83, Rob93].

Diversas categorias de produtos procuram explorar esta nova oportunidade, geralmente em áreas específicas, como por exemplo os da lista abaixo, que está longe de ser exaustiva:

- Editores e planilhas para uso em grupo;
- Vídeo-conferência;
- *Bulletin-Board Systems*;
- Correio eletrônico;
- Sistemas de Workflow.

No presente texto, nos concentraremos no exame dos sistemas de workflow, fazendo referência a produtos de outras categorias para fins de comparação, quando apropriado.

2.4. SISTEMAS DE WORKFLOW

O objetivo dos sistemas de workflow é o de "auxiliar as organizações na especificação, execução, monitoramento e coordenação do fluxo de trabalho em um ambiente de escritório distribuído" [Bul92].

Workflows são normalmente associados a processos das organizações. Um exemplo típico e recorrente na literatura é o do Processamento de Pedidos. Apesar de existirem variações, este processo normalmente corresponde a quatro etapas: 1) a entrada de um pedido, 2) a avaliação do crédito do cliente, 3) expedição de mercadoria e 4) faturamento. Cada uma destas etapas é potencialmente realizada por um funcionário diferente. A entrada de pedido pode ser realizada por um atendente de balcão, a aprovação de crédito por um gerente, a expedição por um encarregado de expedição e o faturamento por um faturista.

É fácil perceber que cada uma destas etapas só pode ser realizada se algumas anteriores, de que dependem, já tiverem sido completadas. A aprovação de crédito, por exemplo, só pode ser realizada após a entrada do pedido. Da mesma forma, só deve ocorrer a expedição e faturamento quando e se o crédito tiver sido aprovado. Algumas atividades podem

ser realizadas em paralelo, caso não apresentem interdependência, como por exemplo, o faturamento e a expedição.

Podemos, a partir deste exemplo simples, determinar as principais funções de um sistema de gerenciamento de workflow: distribuir tarefas a diversos executores diferentes, gerenciando a sincronização entre estas tarefas, de forma que não seja iniciada uma atividade sem que as anteriores de que ela depende tenham sido completadas. Sistemas de workflow são usados em diversos domínios, como o organizacional (ou empresarial), o científico e de engenharia de software, por exemplo. Muitos aspectos são comuns a todos estes domínios, que apresentam porém algumas especificidades. Centraremos nossa discussão em modelos do domínio organizacional.

Sistemas de workflow geralmente estabelecem uma separação clara entre modelagem e execução, como duas etapas distintas. A modelagem identificaria os processos automatizáveis, criando modelos que procuram ser abstrações dos processos de negócio da organização.

Em uma segunda etapa, são criadas instâncias dos processo, chamadas de **casos.** Cada caso flue de acordo com o especificado no modelo do processo correspondente. Estes modelos são interpretados (*enacted*) pelo sistema de workflow, ocasionando o disparo das atividades e a distribuição das mesmas para os agentes cujas descrições constam do modelo (fig. 1).

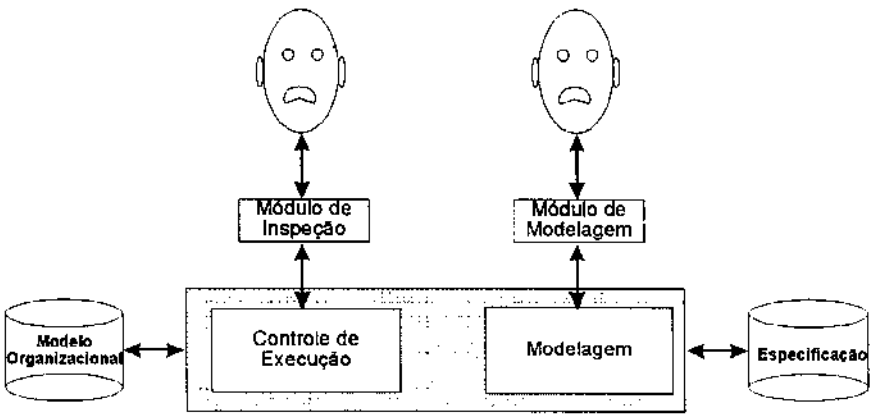


Figura 1 - Arquitetura genérica de um sistema de workflow.

Por trás da aparente simplicidade, se escondem problemas bastante interessantes, que passamos a discutir em seguida.

2.5. O DESAFIO EM SISTEMAS DE WORKFLOW

À primeira vista, a construção dos modelos de processo que são interpretados pelo sistema de workflow pode seguir uma metodologia de construção semelhante à utilizada na determinação de requisitos e modelagem de qualquer outro tipo de sistema convencional: a partir do exame do problema, constrõem-se abstrações que descrevem o processo, que são representadas através de algum formalismo. Este modelo é então interpretado, direcionando a seqüência de disparos de atividade e distribuição de tarefas aos agentes, o que constitui o fluxo de trabalho de cada caso.

A maioria dos sistemas de workflow de primeira geração, construídos durante a década de 70, partiram justamente desta premissa básica, a da semelhança do problema com o já conhecido problema da análise de sistemas. Esta posição foi reforçada pela observação de que por volta de 90% dos processos de uma organização podem ser considerados rotineiros [AGL+93], passíveis em princípio de uma modelagem precisa.

A grande surpresa é de que mesmo em processos aparentemente bem comportados, com alto grau de estruturação, um número significativo de contingências fará com que, na prática, o modelo se torne inútil [Hir86]. Os sistemas de workflow pioneiros provaram ser excessivamente inflexíveis (como relatado em [KHK+91], p.ex.), e a maioria dos projetos da área foram abandonados até a década de 90, que presencia um ressurgimento da pesquisa na área.

A razão do fracasso dos primeiros sistemas parece estar relacionada à variedade de maneiras pelas quais objetivos de negócio das organizações são alcançados, e por características da realidade que se procura abstrair, que resistem a uma modelagem precisa:

- **Organizações diferentes com processos diferentes** - um sistema de workflow deve permitir a representação de uma gama de processos que podem ser muito diferentes, já que estes processos refletem os objetivos de cada organização. Os processos utilizados em uma indústria farmacêutica, por exemplo, podem diferir radicalmente dos utilizados em uma agência de propaganda.
- **Processos semelhantes realizados de maneira diferente** - o modo como os objetivos de negócio nominalmente semelhantes são atingidos em organizações diferentes sofre influência da cultura particular de cada organização. O processamento de pedidos de uma organização pode ser bastante diferente do correspondente em uma outra organização. Não se pode esperar que as organizações se adequem ao modo particular do sistema. Este é que precisa ser suficientemente flexível para se adaptar ao modo específico de cada organização;
- **Mesmos processos feitos de maneira diferente na mesma organização** - da mesma forma que organizações diferentes atingem seus objetivos de maneira potencialmente diferente, dentro de uma mesma organização os mesmos processos podem ser atingidos também de forma diferente, dependendo, por exemplo, da cultura específica de cada filial ou divisão da organização. Esta diferença de abordagem existe em diversos níveis, até a nível de agente: cada um dos agentes pode ter (e provavelmente tem) sua própria maneira de realizar determinadas tarefas. Novamente, não se pode forçar os envolvidos a se adaptarem a um esquema que não lhes seja costumeiro apenas para facilitar a vida dos projetistas dos modelos;
- **Casos específicos tratados de forma especial** - A diferença de tratamento vai ainda mais longe. Um mesmo processo, executado pelos mesmos agentes pode, mesmo assim, não seguir fielmente as etapas previstas no modelo abstrato. Situações especiais, conhecidas na literatura pelo nome (infeliz) de **exceções**, podem fazer com que seja necessário um tratamento especial, feito sob medida para um caso específico. Isto pode acontecer quando existe uma solicitação de urgência de um cliente preferencial, informação incorreta ou incompleta que impede a execução de determinada etapa, quando existe alguma falha de equipamento ou falta de pessoal que obrigue o

redirecionamento de etapas costumeiras, e assim por diante [SM89]. Ao contrário do que o nome implica, o fenômeno das exceções é bastante comum [Hir86, SM89, EN93a, SMS94] e o termo hoje em dia se distancia do seu uso comum, passando a significar apenas ocorrências não previstas no modelo abstrato[5].

- **Processos evoluem** - mesmo admitindo que o modelo representa fielmente uma determinada realidade em relação a um processo, o que, como já vimos, não é verdade, mudanças na realidade externa farão com que os modelos se tornem progressiva ou repentinamente obsoletos. Mudanças de legislação, mudanças de estratégia de mercado e da direção da organização, por exemplo, ocasionam uma impedância entre o representado e a realidade, que exige que se possa executar mudanças dinâmicas [EKR95]. Por mudança dinâmica nos referimos ao fato de que as especificações são modificadas enquanto ainda existem casos do mesmo tipo em andamento. Não se pode simplesmente abandonar os casos existentes, de forma a se poder reiniciar de uma posição segura, devido ao volume de trabalho acumulado que normalmente está associado a estes casos.
- **Processos pouco definidos** - Nem todos os casos seguirão um padrão bem definido, como o de processamento de pedidos, por exemplo. Em algumas situações menos estruturadas, pode-se inclusive ignorar inicialmente qual o processamento a ser aplicado de forma a se obter determinado objetivo de negócio não corriqueiro. É preciso que se possa ir criando a especificação par-e-passo com a execução, ou seja, que não seja necessário se ter um plano completo para se poder utilizar os recursos do sistema de workflow. Deve ser suficiente que se informe, ao final de uma etapa, qual a etapa seguinte (e apenas esta) a ser executada.

Pode-se perguntar como as organizações conseguem atingir seus objetivos, mesmo em presença das incertezas, variações e requisitos dinâmicos que acabamos de expor, em ambientes cujo fluxo não é automatizado. A razão pela qual o trabalho nas organizações é feito (e bem), parece residir na flexibilidade e na capacidade de análise e solução de problemas dos próprios agentes. Segundo Saastamoinen , "o propósito do trabalho nos escritórios não é o de seguir procedimentos, mas pelo contrário, existem certos objetivos a serem atingido e os procedimentos descrevem apenas uma maneira de atingí-los"[6] [Saa94]. O trabalho é, portanto, orientado a objetivos [EW94a] e não conduzido de forma rígida pelas normas, como se poderia supor à primeira vista.

Estas constatações tornam necessária uma mudança de paradigma básico, que deixa de ser o da divisão modelagem/execução, típico da análise de sistemas, em favor a um paradigma mais flexível. Guardadas as devidas proporções, o paradigma mais apropriado parece ser o da planilha de cálculos (citado, por exemplo, em [SMM+94] e [LMY88]). Os principais atrativos de planilhas são:

1) O seu aspecto flexível, capaz de representar uma gama abrangente de instâncias, desde um orçamento doméstico até a planilha de custos de uma grande organização;

---

[5] De qualquer forma, a escolha original deste termo parece expor o pressuposto original dos pioneiros da área, a de que situações deste tipo seriam raras e portanto de menor importância.

[6] *"Offices do not follow procedures as their main purpose. On the contrary, offices have certain goals to obtain and the procedures are only a way to reach their goals"*.

2) São manipuladas diretamente pelos usuários, que são os detentores do conhecimento específico de resolução de problemas;

3) Em uma planilha, a especificação e o uso são feitos de maneira absolutamente uniforme, a ponto de serem indistingüíveis [EM94]: ambas se dão pela digitação de valores e fórmulas em células. Normalmente, os usuários nem se dão conta em que "modo" estão operando, se em modo de especificação ou de execução;

4) Podem ser adaptadas e estendidas de maneira livre, para fazer frente a situações especiais não antecipadas.

Cumpre ressaltar que a flexibilidade, mesmo dos sistemas de workflow mais recentes, está longe de alcançar a flexibilidade das planilhas. Como afirmam Ellis et al., "os sistemas atuais não oferecem os mecanismos necessários para o tratamento de exceções e resolução de problemas em geral"[7] [EN93a].

Salvo pelas linhas gerais acima explicitadas, a semelhança entre sistemas de workflow e planilhas não pode ser levada muito adiante. Nosso trabalho será, então, no restante do presente texto, o de determinar um conjunto de requisitos e conceitos específicos compatíveis em princípio com esta filosofia de flexibilidade enunciada, adequando-a a sistemas de workflow.

Vamos iniciar nosso estudo pelo exame de algumas situações de trabalho [Kyn95], de forma a nos situar mais firmemente no problema de construção de especificações de processo.

2.6. EXEMPLOS DE SITUAÇÕES DE TRABALHO

Apresentaremos a seguir alguns exemplos de situações de trabalho em que sistemas de workflow são ou poderiam ser empregados. A maior parte delas são clássicas, aparecendo de forma recorrente na literatura. Adicionamos a estes alguns outros exemplos menos convencionais que sirvam de estímulo à nossa discussão, demonstrando algumas limitações e problemas relacionados a seu tratamento em um sistema de workflow.

Apresentaremos para cada situação uma descrição, discussão de possíveis problemas e diagramas de representação utilizados em alguns sistemas de workflow, se existentes, à guisa de ilustração.

2.6.1. AQUISIÇÃO DE INSUMOS

Este processo é simétrico ao de Processamento de Pedidos, que discutimos brevemente na introdução do presente capítulo. Este processo descreve os passos de aquisição de material realizados por uma organização. Em linhas gerais, uma solicitação interna de compra é gerada, requisitando alguma espécie de material ou insumo. Seleciona-se em seguida, a partir de um catálogo de fornecedores do produto desejado, um ou mais fornecedores. Dentre estes é selecionado o que oferecer as melhores condições comerciais (preço e entrega), sendo emitida uma ordem de compra, que é enviada ao fornecedor. Quando a mer-

---

[7] *"The mechanisms to help people do their necessary problem solving and exception handling are not available in today's workflow systems".*

cadoria e a fatura referente a esta aquisição são recebidos, é feita a programação do pagamento ao fornecedor.

Alguns pontos podem ser destacados nesta seqüência:

1) Uma vez feita a ordem de compra e enviada ao fornecedor, é preciso aguardar até que a mercadoria e a fatura cheguem. Mesmo que haja uma previsão de data de chegada destes dois componentes, o momento exato é desconhecido, podendo haver antecipação ou atraso. Principalmente no caso de atraso, o sistema deve estar preparado para efetuar alguma ação de notificação que alerte os agentes sobre este fato, de forma que estes possam verificar a causa do atraso e providenciar a sua solução.

2) Como são dois os componentes aguardados, pode acontecer de um deles não chegar (fatura sem mercadoria ou mercadoria sem fatura).

Diversas questões, mesmo neste processo extremamente simplificado, ocasionam problemas de representação em sistemas de workflow existentes:

a) Eventos assíncronos, como o que sinaliza a chegada de mercadoria, normalmente não podem ser representados nas especificações. A tarefa de aguardar a chegada para então se dar prosseguimento ao caso é deixada nas mãos dos agentes;

b) A especificação de ações opcionais, como a de notificação de atraso da mercadoria, associada a um temporizador, são normalmente feitas de uma maneira não integrada com a linguagem de modelagem, o que torna pouco visível a sua existência.

c) Na chegada, a mercadoria e a fatura esperadas tem que ser associadas de alguma forma com o caso onde são aguardadas. Se um grande volume de mercadorias e faturas for recebido, esta associação pode demandar algum esforço, principalmente tendo em vista os tipos diferentes de faturas e notas fiscais utilizados. Este problema é ainda mais agudo se a recepção for feita por pessoal diferente do responsável pelas compras, o que geralmente acontece.

d) Pode-se desejar manter as últimas cotações de material armazenadas, para reutilizá-las, caso estejam dentro de um período de validade. Isto evita que se tenha que executar uma atividade de cotação para cada nova compra, principalmente das mais corriqueiras. A execução opcional de uma atividade baseada no estado dos dados, como a validade das cotações, por exemplo, não é considerada por um grande número de sistemas.

As figuras 2, 3 e 4 apresentam os diagramas referentes a este processo, na visão dos autores e/ou sistemas mencionados nas legendas de cada figura.

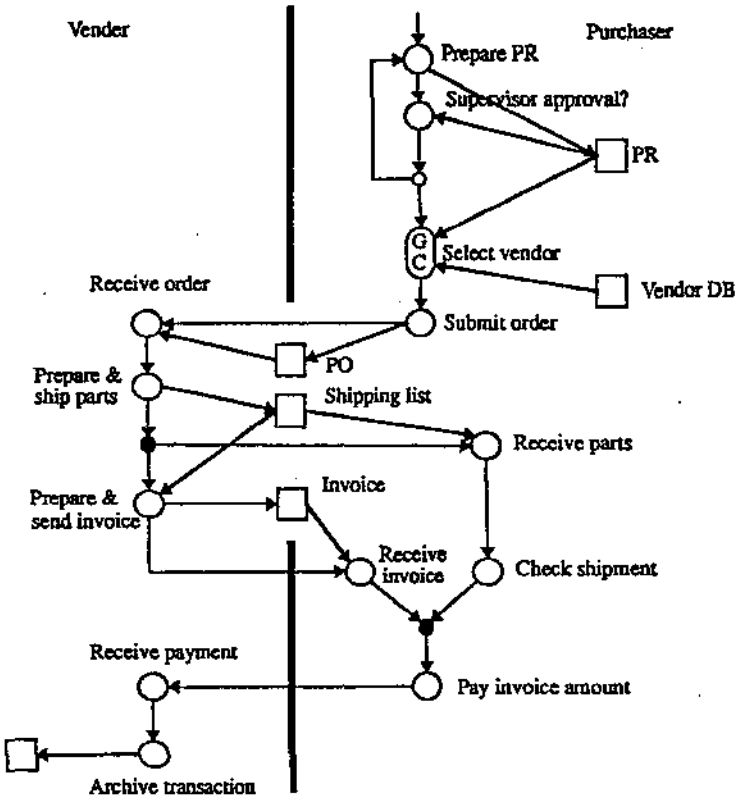


Figura 2 - Ellis e Nutt - ICN [EN93a p.11].

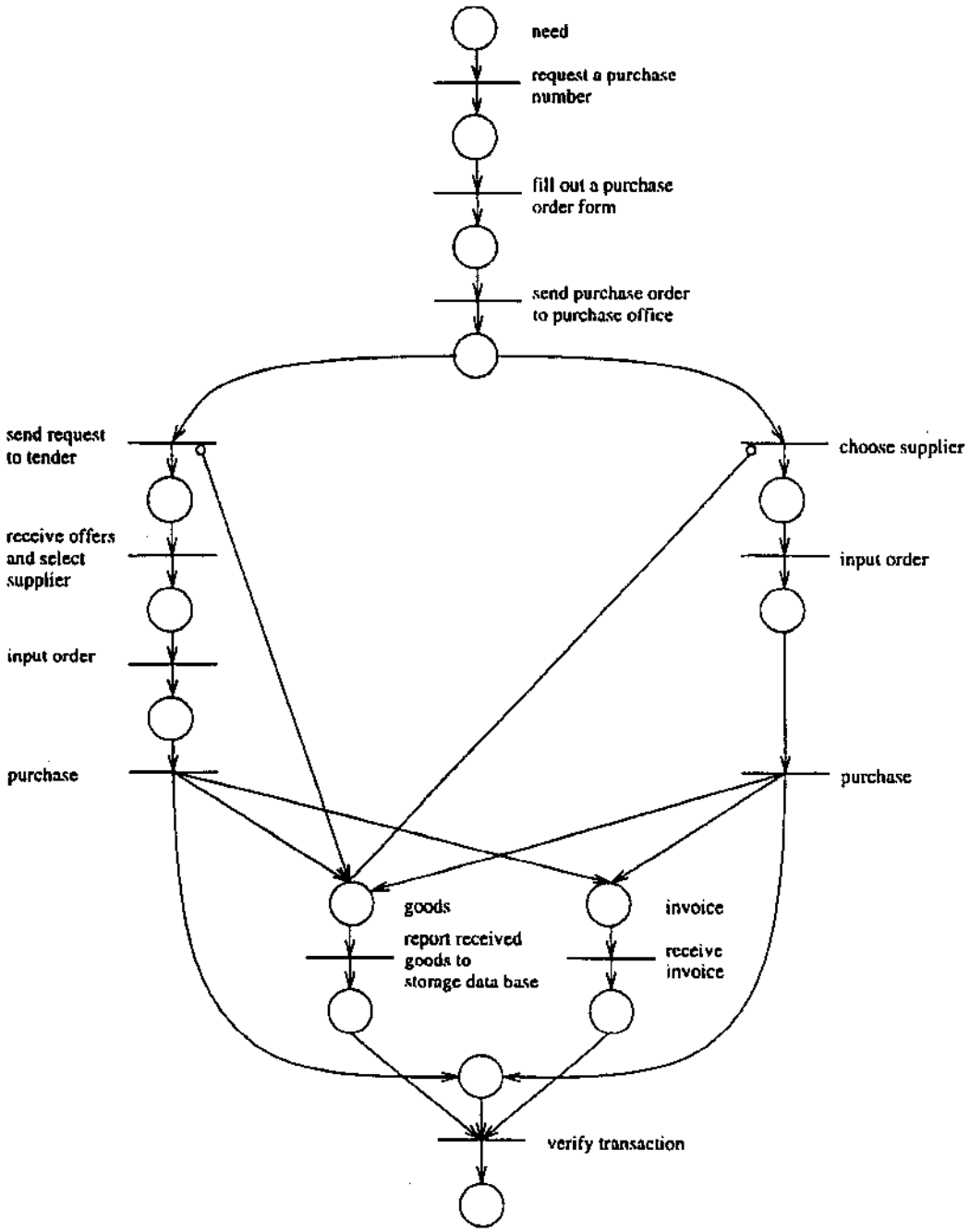


Figura 3 - Saastamoinen and White - [SW95 p.307].

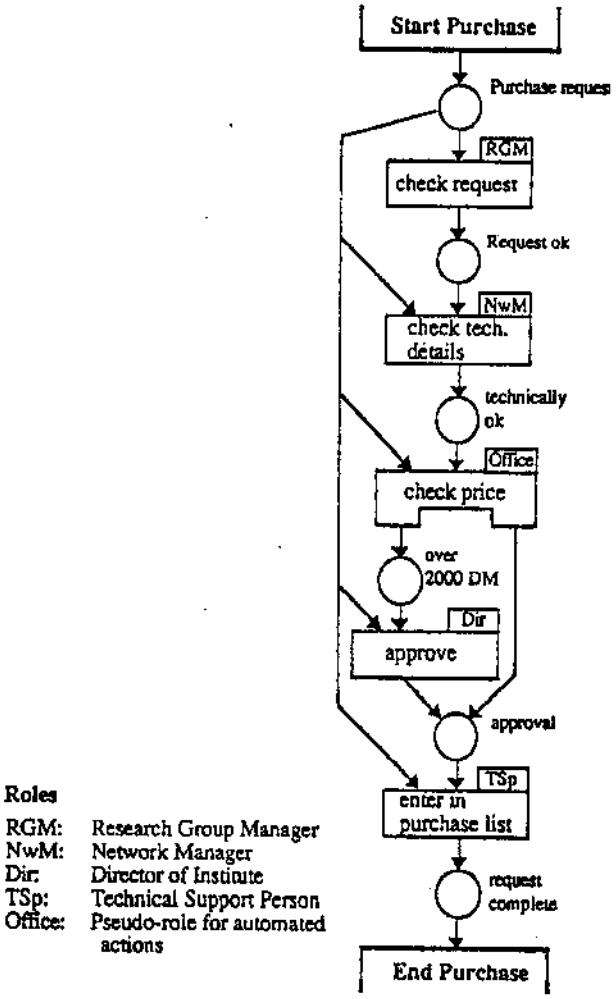


Figura 4 Kreifelts et al. - Sistema Domino [KHK+91 p. 119].

2.6.2. REVISÃO DE ARTIGOS

A revisão de artigos, como acontece na preparação de uma conferência ou de uma revista científica, também é um dos exemplos favoritos da área. A idéia básica é de que diversos artigos são submetidos, sendo analisados independentemente por um determinado número de revisores. Baseado nos laudos de avaliação e possivelmente em outros critérios subjetivos (volume mínimo de artigos necessários, p.ex.), o editor determina se o artigo será ou não aceito para publicação.

Pontos importantes que podem ser destacados:

1) Para cada artigo, a mesma atividade de revisão será executada em paralelo por diversos revisores. Cada um deles produz um laudo que precisa ser consolidado quando todos os revisores tiverem terminado suas avaliações. Eventualmente, pode-se desejar determinar que não são necessários todos os laudos para que se possa avaliar o artigo, bastando que um número mínimo de revisores tenha encerrado sua avaliação.

2) A seleção dos revisores pode seguir critérios subjetivos, como agrupamento por especialidade ou outro semelhante. Cada artigo pode inclusive ser revisado por um número variável de revisores. Um artigo mais complexo ou controvertido, envolverá normalmente um maior número de revisores. Assuntos de extrema especialização poderão ser examinados apenas pelos poucos especialistas disponíveis;

3) Na determinação final de aceitação para o caso de uma *special issue* ou anais de conferência, pode ser necessária a reunião dos dados de todos os artigos. Tendo todos os artigos em mãos, é possível se estabelecer seus méritos relativos por comparação entre as avaliações de cada um deles, além de se considerar outros critérios, como número mínimo e máximo de artigos a serem incluídos, por exemplo.

Várias dificuldades de representação surgem neste processo:

a) Nem todos os sistemas permitem a especificação de atividades replicadas com número variável de instâncias, como pode ser necessário no caso da avaliação precisar envolver um número de revisores apropriado à complexidade do artigo, por exemplo.

b) As vezes, não é possível a seleção de agentes específicos para a execução das atividades de revisão, i.é, o sistema toma para si o encargo de distribuir as tarefas e não admite alternativa. Quando esta atribuição de agentes específicos é permitida, normalmente ela não é explicitada na especificação

c) A reunião de todos os artigos, para que se possa fazer uma análise comparativa entre eles, é especialmente ausente das especificações. Estas atividades, chamadas de **atividades batch** por Barthelmess e Wainer [BW95a], são bastante comuns, mas mesmo assim não existem mecanismos que permitam a especificação de sincronização necessária. O que existe de especial nas atividades batch é que nelas queremos sincronizar **diversos casos**, correspondentes a cada um dos artigos em análise, por exemplo. A totalidade dos sistemas permite apenas que se estabeleça sincronismo relativo a um caso, e não a um grupo de casos relacionados.

As figuras 5 e 6 apresentam algumas versões de representação deste processo.

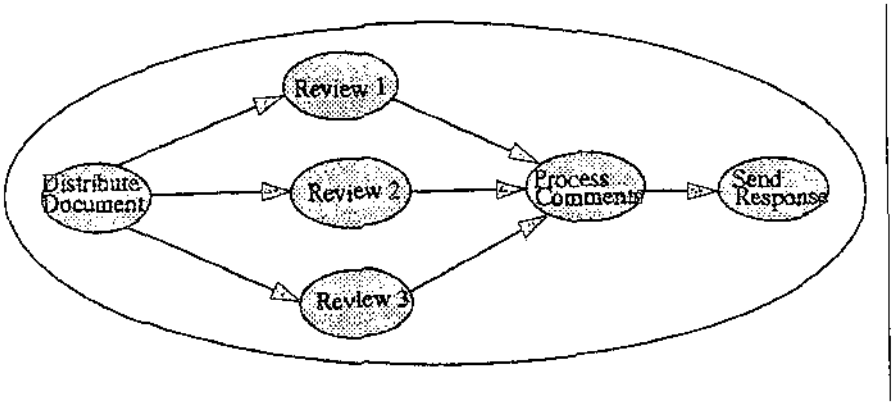


Figura 5 - McCarthy and Sarin - InConcert [MS93 p. 54].

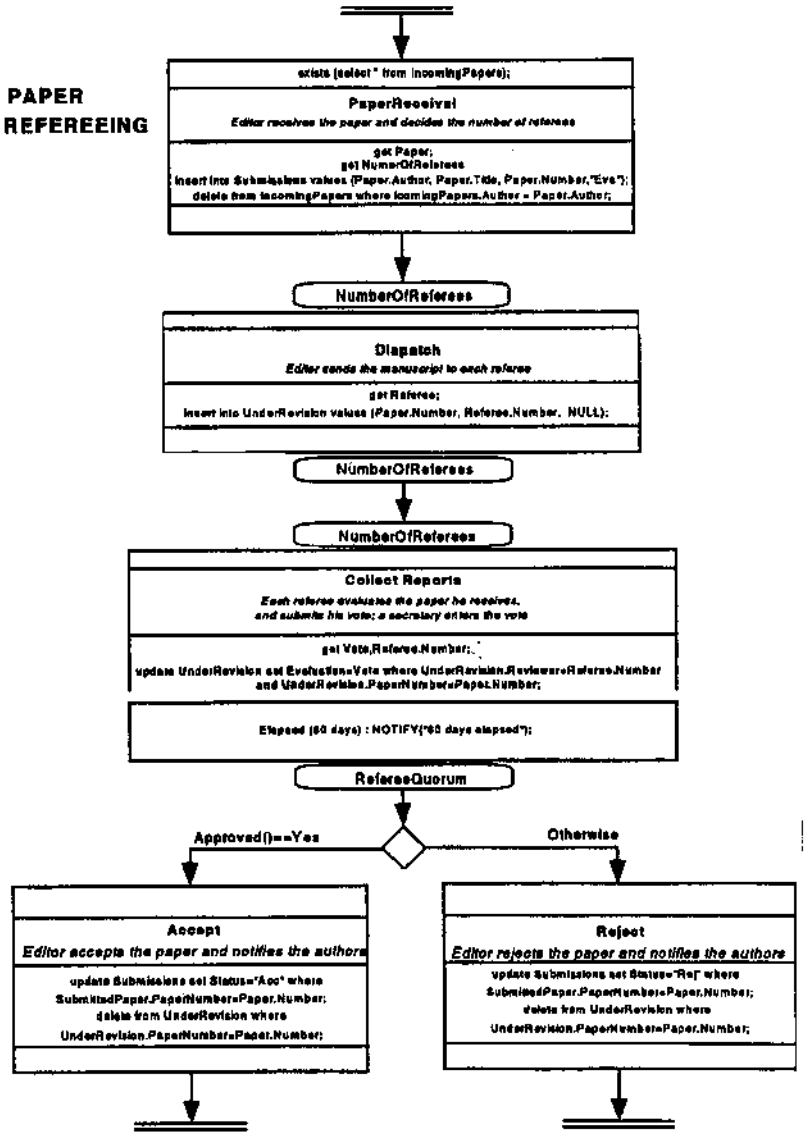


Figura 6 - Casati et al. [CCP+95a p. 11].

2.6.3. SELEÇÃO DE CANDIDATOS À PÓS-GRADUAÇÃO

A seleção de candidatos à pós-graduação ao departamento de ciência da computação da Unicamp (dcc) se realiza anualmente. São recebidas consultas, enviados formulários de inscrição e recebidos ditos formulários acompanhados ou não de outros documentos.

Dos documentos que acompanham a inscrição de cada candidato, alguns necessariamente serão recebidos separadamente do formulário de inscrição: as cartas de recomendação dos professores. Em alguns casos, outros documentos, como currículo escolar, podem ser também recebidos a posteriori.

Os documentos são recebidos e armazenados nas pastas de cada candidato, até que em uma data determinada o coordenador da subcomissão de pós graduação (sub-cpg) inicia o processo de avaliação. A cada candidato são atribuídos revisores, escolhidos dentre os

professores do departamento. São atribuídos pelo menos três revisores para cada candidato. Candidatos ao programa de doutorado são avaliados por cinco revisores. Caso o candidato já apresente um plano de pesquisa definido, o professor responsável pela área pode ser incluído como um dos revisores encarregados da avaliação do candidato, ou como um revisor adicional.

Cada proposta é analisada e cada revisor gera um laudo com informações sobre a análise subjetiva realizada. No final do prazo de avaliação, o coordenador da sub-cpg examina todas as avaliações de cada candidato, classificando-os de acordo com os critérios expressos pelos revisores. Um determinado número dos candidatos com melhor classificação serão aceitos no programa, e os demais rejeitados. Em todos os casos, uma carta é enviada aos candidatos, informando-os do resultado.

A relação dos aprovados é utilizada em outro contexto, o de inscrição, a ser iniciado posteriormente.

Este exemplo simples apresenta diversas características que o tornam interessante do ponto de vista de workflows:

1) Documentos complementares chegam de forma assíncrona, i.é, não se pode saber quando (e se) as cartas de recomendação e o currículo de cada candidato serão recebidos;

2) Na atribuição de revisores e principalmente na classificação, dados de todos os candidatos precisam estar reunidos para sofrerem um processamento conjunto. Estas são, novamente, atividades batch [BW95a], nas quais não se deseja tratar cada caso individualmente, mas sim o conjunto de casos como um todo.

3) No momento da distribuição de candidatos entre os avaliadores, diversos critérios podem ser empregados, como agrupamento por instituição de origem, de acordo com as áreas de interesse declaradas, e assim por diante. Novamente, esta é uma atividade batch, que se beneficia da reunião de todos os casos para que se possa aplicar uma otimização qualquer.

4) Em vários momentos, na avaliação realizada por cada professor e mesmo na classificação final, fatores subjetivos não estruturados estão em jogo.

5) A avaliação é especialmente interessante, pois pode ser realizada de muitas formas diferentes, agrupando-se ou não as informações dos candidatos e realizando-se a avaliação de forma individual, em grupos com um número determinado de participantes (professores formando uma banca, p.ex.) ou coletivamente por todos os professores. Se a atividade for individual, cada revisor recebe um determinado número de candidatos para avaliar, executando seu julgamento de forma desvinculada dos demais. Esta é a única forma disponível na maioria dos sistemas existentes. Pode acontecer que se decida instituir bancas de avaliação, de forma que grupos de revisores examinam e julgam em conjunto os candidatos sob sua responsabilidade. Esta atividade coletiva pode ser levada ainda mais adiante, quando então todos os revisores formam um único grupo, responsável por todos os candidatos, sem subdivisões explícitas de responsabilidade.

Esta atividade pode ser feita de três maneiras alternativas:

- Em isolamento - cada agente tem acesso apenas a seus próprios laudos de avaliação, não podendo verificar a opinião dos demais revisores;

- Com acesso aos laudos - neste caso, os laudos dos demais revisores podem ser examinados pelos demais, havendo um compartilhamento das opiniões. Cada revisor continua a produzir o seu próprio laudo, mesmo que influenciado pela opinião dos demais;
- Banca de avaliação - os revisores trabalham em conjunto, produzindo um laudo único, que reflete a posição conjunta de todos os revisores.

Atividades coletivas deste tipo não são incomuns, e apresentam requisitos de suporte bastante diferentes das atividades individuais, principalmente as relacionadas à comunicação.

A aparente simplicidade deste processo apresenta armadilhas que fazem com que sejam não implementável na sua totalidade por praticamente nenhum sistema de workflow existente:

a) A distribuição de documentos que chegam assincronamente não é facilmente tratada em muitos modelos. Especialmente importantes são as contingências que este tipo de situação acarreta, como a chegada de cartas de recomendação sem o formulário de inscrição, e vice-versa, ou chegada parcial dos documentos.

b) A especificação da atividade de classificação é especialmente difícil. Nesta atividade, os dados de todos os candidatos devem estar necessariamente reunidos, pois envolve comparações entre todos eles. Esta é mais uma vez uma atividade batch, como a que examinamos no exemplo de revisão de artigos.

c) O suporte efetivo à realização de tarefas subjetivas envolve a possibilidade de se oferecer subsídios alternativos, dependente de critérios desconhecidos do sistema e que muitas vezes são função do executor da tarefa. Na avaliação dos candidatos, por exemplo, cada revisor pode ter seu método próprio de análise, necessitando de informações diferenciadas para realizá-la (histórico de candidatos da mesma instituição, indicados pelos mesmos professores, etc.). A maioria dos sistemas apresenta pouca ou nenhuma preocupação com ambientes de execução que facilitem o trabalho dos agentes.

d) A determinação dos executores, segundo critérios subjetivos, necessária na atividade de distribuição, normalmente não é diretamente representável na maioria dos sistemas, como já mencionamos em relação ao processo de revisão de artigos.

e) Atividades coletivas são difíceis de representar, não sendo possível realizar a avaliação através da discussão conjunta de diversos participantes, mesmo que a comunicação não seja síncrona.

f) A comunicação permitida pelos sistemas de workflow costuma se restringir àquela formalmente presente no modelo. Consultas espontâneas, não previstas na especificação normalmente não são permitidas, ou são suportadas por mecanismos rudimentares. Isto impede, por exemplo, que um professor consulte um colega a respeito de um determinado candidato. Se, durante a fase de classificação o coordenador precisar consultar algum dos revisores em relação a um laudo pouco claro, esta ação também será de difícil execução na maioria dos sistemas.

2.6.4. CONTROLE DE TRÁFEGO AÉREO

Apesar de não constituir um workflow no sentido estrito do termo, já que na descrição do trabalho na sala de controle de tráfego aéreo que examinaremos a seguir não é explicitada a sub-divisão em atividades, este exemplo é representativo de uma certa corrente da área de CSCW, sobre a qual discutirmos em 3.2.-SOBRE O TRABALHO.

Apesar de ser possível em princípio se descrever este trabalho em termos de etapas realizadas por cada um dos controladores, o interesse principal da descrição que examinaremos a seguir é justamente o de abordar o problema pelo ângulo do coletivo, e não do individual. A descrição abaixo é adaptada a partir dos relatos de [BR94] e [Rob93]:

O controle de tráfego aéreo é feito através da anotação dos dados de vôo em tirinhas (*flight strips*). Estas tirinhas contém informações sobre o estado atual de vôos sendo controlados, bem como o estado futuro esperado, além de instruções dadas à aeronave pelo controlador responsável. As informações são criadas inicialmente a partir de um banco de planos de vôo especificados pelos pilotos antes da partida, e contém o número de identificação do vôo, a altura e direção atuais, rota planejada, pontos de controle de navegação nesta rota, tempo estimado de chegada, aeroportos de origem e tipo da aeronave (fig. 7).

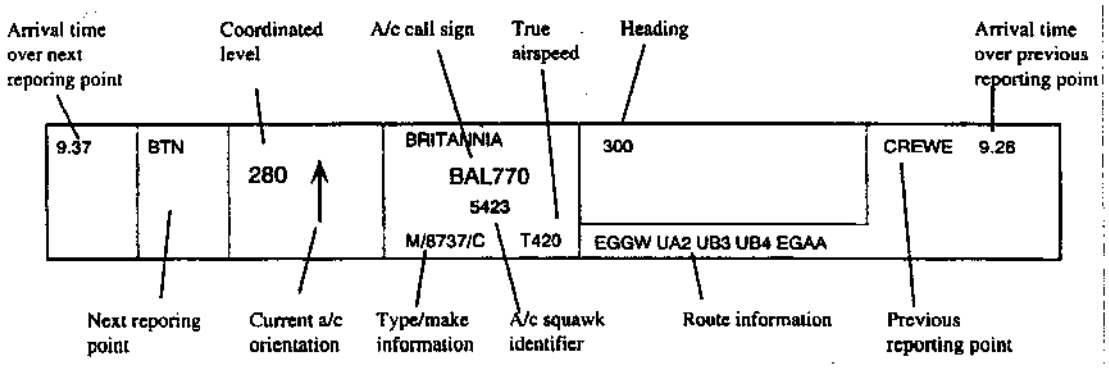


Figura 7 - Tira de controle de vôo [BR94 p.4].

Estas tirinhas são dispostas em um painel de progresso de vôo (*flight progress board*), organizadas em um *rack* colocado diretamente sobre as telas dos radares, de acordo com os pontos de controle sobre os quais os vôos irão passar. Este painel serve como um ponto central de coordenação para os controladores de vôo, que com uma olhada conseguem prever, por exemplo, quantas aeronaves devem chegar ao setor, em que momento e quais os seus destinos. Isto permite um diagnóstico precoce de sobrecargas (muitos vôos chegando ao mesmo lugar em tempos próximos). Conforme o vôo progride, as instruções passadas pelos controladores e confirmadas pelos pilotos são anotadas nas tiras. Estas marcas obedecem a uma convenções compartilhadas por todos os operadores: se foi transmitida a ordem para que a altitude fosse aumentada para o nível 220, a tira será anotada com uma seta para cima seguida do número do nível, 220. Assim que o piloto confirma a nova altitude, a altitude anterior é riscada na tira. Qualquer outra modificação de direção, tempo estimado de chegada, etc. é anotada de maneira similar.

Na maior parte do tempo os controles parecem estar trabalhando em isolamento, cada qual operando com os vôos referentes à sua área de responsabilidade. A manutenção do painel permite, porém, que o estado global seja conhecido por todos os controladores.

Esta consciência constante do estado geral propicia que, conforme percebem através do exame do painel que um colega está sobrecarregado, os controladores possam começar a auxiliá-lo automaticamente, sem que uma delegação explícita de tarefas tenha que ser feita.

No final do dia, a série de tiras e a organização resultante das modificações incrementais realizadas por cada controlador servem como resumo das atividades realizadas, incorporando a história do setor pelo qual ele é responsável. Cada controlador usa uma caneta de cor diferente para fazer as anotações, de forma que se sabe a qualquer momento quem é o responsável por elas. Qualquer outro controlador treinado pode ler neste painel, com um olhar apenas, qual foi a seqüência de eventos transcorridos e quem fez quais modificações.

As tiras são ordenadas nos *racks* por tempo previsto de chegada do vôo ao ponto de controle. Parece natural que as tiras pudessem ser inseridas automaticamente no lugar correto, caso se resolvesse automatizar este processo. Surpreendentemente, porém, observa-se que o posicionamento manual tem uma função fundamental, a de chamar a atenção dos outros operadores para este novo evento, auxiliando-os a estarem atentos a problemas potenciais que possam vir a acontecer em decorrência deste fato novo.

Um outro mecanismo utilizado para a divulgação implícita de problemas potenciais é a de que, assim que um operador qualquer percebe um problema com dois ou mais vôos (provavelmente durante a inserção de uma nova tira), ele levanta levemente as tiras correspondentes, desalinhando-as em relação às demais. Para o olho treinado dos controladores, não só o ponto de problema se torna aparente, mas também a razão (graças à convenção de anotações).

Podemos destacar alguns pontos nesta discussão:

1) Quem sabe a conclusão mais importante é a de que o trabalho não precisa ser necessariamente descrito como uma série de etapas sucessivas, como nos exemplos anteriores de processo que examinamos.

2) As observações revelam uma riqueza de detalhes que normalmente não é encontrada nas descrições de processos. Isto decorre, possivelmente, na ênfase dada à interação dos agentes, em detrimento da busca de uma abstração generalizante.

3) Este exemplo destaca a importância dos artefatos na realização de trabalho coletivo, como mediador da ação dos agentes. O artefato tem uma organização e regras de uso convencionadas entre os agentes, e transmite um grande volume de informações de forma implícita.

Podemos identificar duas dificuldades principais relacionadas a este exemplo, se desejássemos incluí-lo em uma especificação de workflow:

a) Atividades coletivas não são possíveis na maioria dos sistemas de workflow. Estes pressupõem normalmente que as atividades são subdivididas repetidamente até que se obtenha atividades individuais. Esta subdivisão, no caso analisado, inviabiliza o trabalho, já que dificulta a comunicação entre os agentes.

b) Normalmente, os objetos de suporte ao trabalho em si são considerados como fora do escopo do sistema de workflow em si. Este normalmente se preocupa apenas com a sincronização e distribuição de tarefas, essencialmente ignorando a realização do trabalho

em si, como atesta a afirmação de Winograd, que adaptamos: "a efetiva execução de seja lá o que for que é necessário para se atingir o objetivo [da atividade] está fora do escopo [do sistema]" [Win86 p. 207]. Isto ilustra um ponto que iremos enfatizar no capítulo 3, o do radicalismo de certas posturas adotadas por sistemas nesta área.

É importante notar que neste exemplo em especial, a complexidade não se encontra na subdivisão clara das etapas individuais, mas em um uso altamente estruturado de um artefato, o rack onde as tiras são armazenadas. As anotações e a própria movimentação das tiras é que apresentam uma estruturação rígida de uso, e não a seqüência de etapas de sua manipulação.

2.6.5. Criação de um novo produto

Este processo, devido a Swenson et al. [SMM+94], é iniciado por um pedido genérico proveniente da presidência da organização, endereçado à divisão de desenvolvimento de produtos da organização.

O atrativo fundamental deste processo consiste justamente na sua generalidade radical, o que torna a construção de uma especificação a *priori* se não impossível, bastante difícil. A variedade de possibilidades de condução deste processo, faz com que as etapas, especialmente no início do processo, tenham que ser desenvolvidas à medida em que são necessárias ("*on-the-fly*"). Conforme o caso progride, será eventualmente possível se estabelecer etapas mais padronizadas, para execução por pessoal de linha de frente, os projetistas em si, pessoal de marketing e assim por diante.

Este exemplo é representativo de uma classe de processos onde se conhece de início apenas um objetivo genérico. A determinação do modo pelo qual este objetivo será alcançado é parte essencial do trabalho relacionado a ele. Em outras palavras, o planejamento das etapas representa uma grande parte do trabalho envolvido em se alcançar o objetivo, o que normalmente não acontece com processos mais corriqueiros, onde o trabalho se concentra na execução das tarefas para as quais já existe uma subdivisão em etapas. Note que neste tipo de processo é virtualmente impossível se utilizar a separação modelagem/execução de forma clara, já que ambas evoluem em conjunto, conforme o caso progride.

2.7. Conclusões

Sistemas de workflow se inserem no contexto geral de software cujo objetivo é o suporte ao trabalho cooperativo, onde se enfatiza a interação entre usuários, e não apenas a interação usuário/sistema.

O que torna a área interessante é a grande variação apresentada na prática pelos processos, devida a diferenças de cultura empresarial, de grupo e de agente, exacerbada pela existência de casos especiais e pela evolução natural dos processos no decorrer do tempo. Em decorrência disto, novos paradigmas devem ser utilizados, em detrimento da divisão tradicional entre modelagem e execução. Um paradigma mais apropriado, ressalvadas as diferenças, é a das planilhas de cálculo, onde modelagem e execução são feitas de maneira absolutamente uniforme.

Os exemplos apresentados mostram que existem problemas de representação, mesmo em processos tão simples como os apresentados na seção 2.6.

3. FUNDAMENTOS DO PROBLEMA

3.1. INTRODUÇÃO

No capítulo anterior, caracterizamos os sistemas de workflow, mostrando que estes se inserem em um contexto de software voltado para o trabalho cooperativo, e identificamos características que os tornam especiais, diferenciando-os dos sistemas convencionais. Analisaremos agora os fundamentos associados a sistemas deste tipo, de uma forma distanciada dos detalhes. O que queremos neste momento é obter uma visão geral da questão, identificando problemas estruturais nas soluções adotadas por sistemas de workflow atuais.

A partir de uma discussão sobre o trabalho (seção 3.2), elementos fundamentais (3.3) e sobre o papel das ações *ad-hoc* e especificações (3.4), identificaremos tendências gerais e mostraremos que as soluções atuais cobrem apenas parcialmente o espectro de possibilidades, resultando em soluções parciais para o problema. Chamaremos a atenção especialmente para o fato de que as categorias de sistemas de CSCW parecem escolher soluções extremas dentro do espectro identificado, que podemos identificar com duas correntes filosóficas antagônicas, que discutiremos em 3.5.-VISÕES DE MUNDO.

3.2. SOBRE O TRABALHO

O fracasso dos sistemas pioneiros de workflow pode ser explicado em parte por observações de autores ligados a uma corrente da antropologia, a etnografia. A partir de observações normalmente realizadas por equipes multidisciplinares em ambientes reais de trabalho, autores, entre os quais se destacam Robinson, Bannon, Schmidt e Suchman, revelam a complexidade insuspeitada do modo como o trabalho das organizações parece ser efetivamente conduzido.

Apesar do radicalismo ocasional de certas posições destes autores, estes comentários são fundamentais para que se possa chegar mais a fundo nos fatores que determinam a aceitação e sucesso de um sistema de workflow, nos fornecendo principalmente uma série de termos que utilizaremos no decorrer do restante da nossa discussão.

- **Ação situada** - este termo se refere ao argumento de que nenhuma ação faz sentido fora do contexto em que é tomada. Suchman alerta que "problemas surgem quando representações normativas são geradas à distância dos locais onde o trabalho que procuram representar é efetivamente realizado, ou quando estas mesmas representações são utilizadas em substituição ao trabalho propriamente dito"[8] [Suc95]. As representações não podem, portanto, ser consideradas como uma descrição isenta da Verdade [Rob93, EW94a]. A posição destes autores é complementada pela observação de Swenson et al. de que as descrições do trabalho obtidas em entrevistas costumam não representar o modo pelo qual o trabalho é efetivamente realizado [SMM+94];

[8] *"Problems arise when normative representations are either generated at a distance from the sites in wich the work they represent goes on, or are taken away from those sites and used in place of the work itself"*

- **Trabalho de articulação** - o trabalho de articulação é aquele necessário para que os agentes envolvidos em uma tarefa comum alinhem seus pontos de vista e estabeleçam um consenso sobre o uso e interpretações a serem adotadas. Um exemplo deste tipo de trabalho é relatado por Iochpe em [Ioc95], em relação a conflitos de interpretação na fase de análise de requisitos durante o desenvolvimento de software. Iochpe propõe um sistema interessante para resolução destes conflitos, que pode ser visto como um facilitador do trabalho de articulação entre pessoal técnico e usuários consumidores, permitindo que estes cheguem a um consenso em relação à importância e significado de termos utilizados na descrição de requisitos de sistemas;
- **Transições fluidas** - se refere ao fato de que o trabalho transita sem sobressaltos, sem noção ou respeito a categorias que possam ser estabelecidas. Uma fronteira em especial costuma não ser apropriada, a da modelagem e execução. Esta fronteira, bastante natural no desenvolvimento de sistemas automatizados em geral, parece não ser apropriada no caso de sistemas de workflow. A diversidade de casos ocasiona a necessidade de se entremear a execução com a especificação, para que se possa redirecionar cada caso particular de acordo com as suas necessidades específicas. O termo se refere igualmente a outras dimensões, como o individual e o coletivo, por exemplo: uma atividade em princípio individual pode vir a se tornar coletiva, caso seja necessário envolver outros agentes na resolução de um problema mais complexo, por exemplo;
- **Uso não antecipado** - parece constituir uma forte característica o fato de que na maioria dos sistemas o uso dado pelos usuários não coincide com a intenção expressa ou implícita de seus projetistas. Um exemplo disto é citado por Robinson em [Rob93], a respeito do sistema The Coordinator, que é utilizado na prática de maneira bastante diferente do que o pretendido pelos seus projetistas. Este tipo alternativo de uso deve ser incentivado, através do projeto dos próprios sistemas, que devem, dentro do possível, assumir uma posição neutra que permita este tipo de utilização inovadora.
- *Awareness* (ciência) - o trabalho de cada agente não é feito em absoluto isolamento, mas precisa existir uma sinergia entre eles. A *awareness* diz respeito à difusão de informação de contexto para todos os participantes de alguma atividade, eliminando o senso de isolamento que pode surgir do uso de um sistema automatizado estanque [KHK+91]. Todo agente precisa estar ciente da razão da sua atividade, de onde ela se encaixa no panorama geral do problema, quem são os outros agentes e o que eles estão fazendo ou já fizeram em relação à atividade sendo executada. A necessidade deste tipo de mecanismo vem da observação de que o trabalho de um agente é direcionado pelas ações dos seus colegas. É justamente isto que torna o trabalho colaborativo, fruto de um esforço coletivo que é diferente da soma dos esforços individuais isolados;
- **Duplo nível de linguagem** (*"double-level language"*) - as observações demonstram ainda que existe, além da comunicação explícita entre os agentes, um segundo nível, de comunicação implícita, que é transmitido através dos artefatos utilizados como ferramentas de trabalho [Rob93]. Robinson utiliza o exemplo do quadro de chaves de um hotel que, com um simples olhar, pode indicar ao agente treinado uma série de informações (quais os hóspedes que se encontram em seus aposentos, existência de

mensagens e assim por diante). Um exemplo mais próximo da realidade das organizações são os próprios formulários. Um formulário é projetado de forma a capturar um conjunto de informações que flui entre os agentes sem que estes precisem se comunicar explicitamente sobre elas, a não ser em casos excepcionais.

As posições parecem convergir em direção à utilização de artefatos comunitários ("*common artifacts*") [Rob93] como meio de construção de ferramentas de suporte ao trabalho cooperativo. O artefato comunitário funciona como um espaço comum através do qual o trabalho é tornado explícito, como em um quadro de chaves de hotel, ou o painel de controle utilizado no controle de tráfego aéreo (veja o exemplo descrito no início do presente texto). Um artefato incorpora muitos dos conceitos apresentados, sendo utilizado para suporte ao **trabalho situado**, propiciando **transições fluidas**, permitindo o **uso não antecipado** e servindo de mecanismo de difusão de ***awareness***. O seu uso pressupõe um **duplo nível de linguagem**, o implícito e o explícito, quando os agentes conversam sobre o artefato durante o **trabalho de articulação**:

- Artefatos são previsíveis - possuem uma estrutura e um comportamento conhecido e estável, com o qual todos podem contar;
- São fruto de um processo evolutivo - não existe uma regra mágica que diga como um artefato deve ser construído, mas versões de artefatos são depuradas durante o uso, de forma a tornarem fácil a resolução dos problemas principais;
- Propiciam comunicação implícita - ações realizadas sobre o artefato comunicam de forma indireta, sem necessidade de troca de explícitas de mensagens. Cabe ressaltar que a existência da comunicação indireta não indica que se pode prescindir da comunicação direta, mas sim que os dois níveis são necessários ("*double-level language*");
- Funcionam como contexto de discussão - o artefato serve como mecanismo de mediação do movimento dialético de resolução de um problema. O artefato fornece uma referência comum em torno da qual as discussões giram, como por exemplo um gráfico ou diagrama que é comentado, anotado, referenciado e modificado por cada um dos participantes de uma discussão como modo de tornarem seus pontos de vistas conhecidos;

O resumo apresentado acima não substitui a leitura dos próprios textos, ricos principalmente na descrição de situações reais de trabalho, e que recomendamos aos leitores. Alguns exemplos são: a sala de controle de metrô [Rob93], controle de trânsito de aeronaves [Rob93], divisão de instalação de companhia telefônica [Sac95], empresa de consultoria [Orl92], escritórios de contabilidade [Suc83], de advocacia [Suc95], agência de viagens [Rog94], além do controle de tráfego aéreo [BR94, Rob93], descrito como um dos nossos exemplos de situações de trabalho (2.6.4.-CONTROLE DE TRÁFEGO AÉREO).

3.3. ELEMENTOS FUNDAMENTAIS

Ellis e Wainer estabelecem em [EW94a] uma taxionomia de componentes de groupware, elucidativa quanto aos elementos fundamentais deste tipo de sistema. São propostos os paradigmas do **sincronizador**, **guardião** (*"keeper"*) e **comunicador**[9].

- Como o nome indica, o **sincronizador** se preocupa com a sincronização das ações realizadas por um grupo de agentes. A principal preocupação é controlar o início de execução das atividades de forma que estas só aconteçam quando os subsídios necessários, frutos de atividades precedentes, já estiverem disponíveis. Ferramentas para controle de projetos, como PERT, e sistemas de workflow são exemplos desta categoria de groupware.

  Este aspecto está relacionado ao estabelecimento claro de etapas de trabalho, das interdependências entre estas etapas e ao suporte à sua distribuição entre os agentes.

- A função de **guardião** (do artefato) é exercida pelos componentes que atuam como repositórios e propiciam um acesso controlado ao artefato sob a sua guarda. Um sistema de CAD, sistemas CASE, ou um editor de textos colaborativos (multiusuário síncrono), como Grove [EGR90] são alguns sistemas predominantemente deste tipo. A ênfase deste aspecto é, portanto, o gerenciamento de objetos.

  A guarda de artefatos pressupõe que o sistema ofereça recursos poderosos de gerenciamento e compartilhamento de objetos, que aumente a sinergia entre os participantes [EW94b], suporte a versões [EKR95, AS94], limitação de acesso [BR94] e interface colaborativa que ofereça mecanismos de difusão de ações realizadas sincronamente por outros agentes sobre objetos compartilhados [EGR90].

- **Comunicadores** facilitam a comunicação entre diversos agentes. O principal exemplo desta categoria são os sistemas de correio eletrônico, refletindo a predominância neste aspecto, da troca de informações entre os agentes.

  Os aspectos envolvidos incluem a comunicação assíncrona, através de mensagens, e/ou comunicação síncrona, através do uso de recursos de vídeo e voz em tempo real.

Uma observação importante é a de que sistemas colaborativos existentes costumam se alinhar de acordo com certos padrões radicais dentro do espectro de possibilidades, com a predominância de um dos três aspectos, como se pode perceber nos exemplos associados a cada um dos aspectos descritos acima.

Representaremos a influência dos três aspectos através de um gráfico de barras, conforme ilustrado pela figura 8, que representa a importância relativa dos três aspectos em sistemas de workflow. Neste gráfico, quanto maior a barra, maior a influência do aspecto correspondente. Como visa representar um conjunto de sistemas, a proporção entre as barras não corresponde a um parâmetro exato, mas relativo.

---

[9] Ellis adiciona um quarto paradigma, o de agente, associado a sistemas especialistas que funcionam como conselheiros sobre aspectos específicos do trabalho (a colocação de equipamento em uma cozinha, p.ex., [FNO+93]). Esta categoria de componentes é geralmente utilizada em conjunto com outros sistemas mais abrangentes, não configurando, em nosso entender, um elemento básico por si.

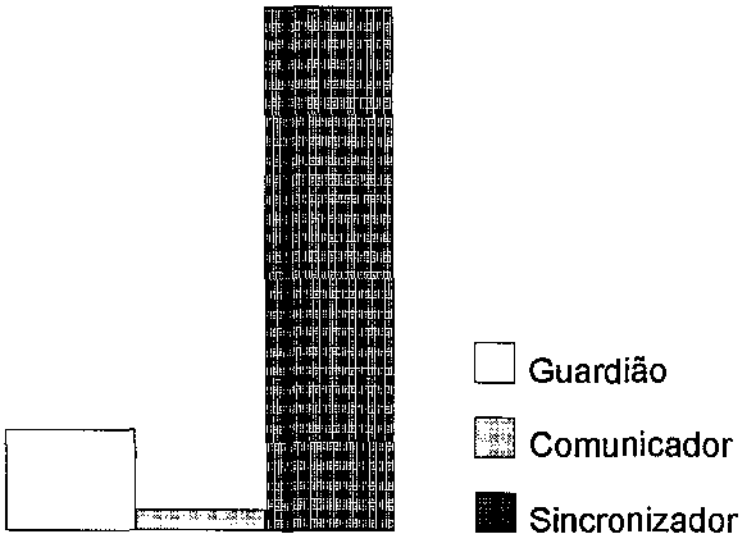


Figura 8 - Sistemas de workflow.

Nos sistemas de workflow, a influência predominante é claramente do elemento sincronizador. Nestes sistemas, as etapas de trabalho e a sua distribuição para os agentes apropriados é o tema central, em detrimento dos demais aspectos. A guarda dos artefatos tem uma influência secundária, representada pela manipulação dos documentos associados aos casos, enquanto que a comunicação é normalmente limitada. A necessidade de se distribuir as tarefas faz com que estes sistemas tenham maior informação sobre a estrutura organizacional em que se inserem, como as unidades organizacionais, a hierarquia entre agentes e assim por diante.

A ênfase é dada ao processo visto como sequenciamento de atividades individuais, com uma forte tendência taylorista. O trabalho, em princípio coletivo, é dividido em tarefas estanques, que são atribuídas a agentes, que em alguns casos ignoram a própria origem e o destino do trabalho que agregam [KHK+91].

Vamos examinar agora algumas outras classes de sistemas da área de CSCW, onde predomina algum dos outros aspectos.

Em editores e planilhas colaborativas, a ênfase é claramente o elemento de guarda de artefatos. Os mecanismos associados à gerência dos objetos permite que estes sejam utilizados de forma colaborativa, por mais de um agente simultaneamente, se encarregando de difundir as ações de cada um deles para os demais. Esta difusão implica necessariamente na existência de um componente comunicador relativamente forte. Supõem-se que os agentes são os responsáveis por se auto gerirem, distribuindo as tarefas entre si sem apoio do sistema, o que corresponde a um aspecto sincronizador inexistente (fig. 9).

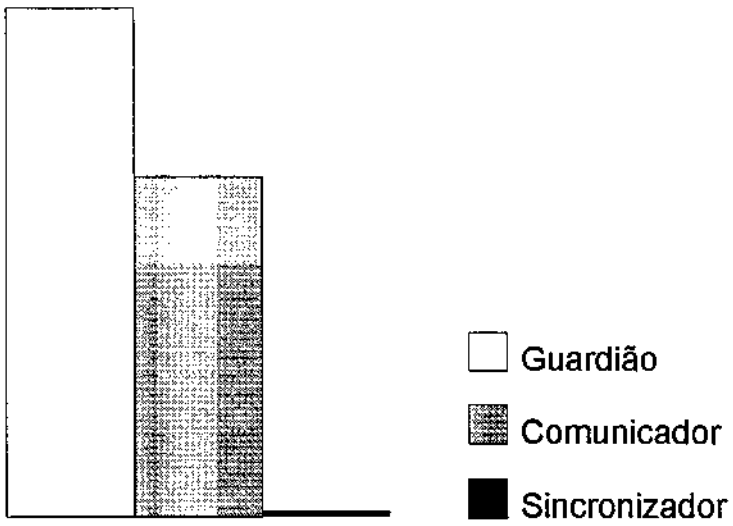


Figura 9 - Editores e planilhas colaborativas.

Na categoria dos produtos de suporte à tele-conferência, a ênfase não poderia deixar de ser a comunicação. O objetivo primário é o de permitir contato síncrono entre os agentes, que podem fazer uso de artefatos como trechos de planilha, textos e gráficos como meio para exporem suas idéias, o que corresponde a um uso específico do aspecto guardião. Novamente, o aspecto sincronizador é ignorado, ficando a cargo dos próprios agentes a condução do trabalho e o papel desempenhado por cada um deles.

Podemos distinguir dois tipos de tele-conferência: a vídeo-conferência e a texto-conferência. Na vídeo-conferência ou reunião síncrona à distância, os participantes, que estão em locais diferentes, interagem basicamente através de câmeras de vídeo. O principal uso destas câmeras é o de exibir desenhos, diagramas e anotações que cada agente faz em seu próprio escritório [EW94a]. Na texto-conferência ou suporte a reuniões síncronas co-localizadas, diversos agentes se reúnem na mesma sala para uma tomada de decisão conjunta. O sistema é utilizado para difundir, a partir das estações individuais de cada participante, informações como gráficos e planilhas, em um painel conjunto. Os gráficos referentes a cada um dos modos de tele-conferência são apresentados na figura 10.

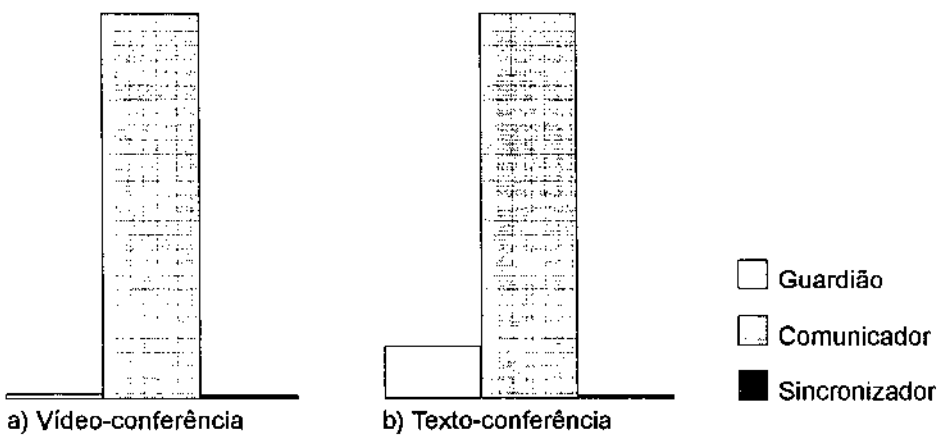


Figura 10 - Tele-conferência.

Apesar de existirem alguns sistemas que apresentam uma influência um tanto mais equilibrada em relação aos aspectos básicos, estes sistemas se concentram na resolução de problemas específicos, como o do processo de elaboração e revisão de documentos provido pelo ForComment [For89], por exemplo (fig. 11), que permite que um texto transite entre diversos agentes, em etapas pré-definidas fixas (elaboração, revisão, modificação). Sistemas de uso mais geral acabam se agrupando em posições extremas do espectro de possibilidades, conforme observação de Ellis e Wainer em [EW94b]: sistemas de estágio único ("*single stage*"), sem divisão em sub-atividades, apresentam aspecto guardião forte, enquanto que sistemas com aspecto sincronizador poderoso ("*multi stage*"), como os sistemas de workflow, tendem a apresentar aspecto guardião simplificado.

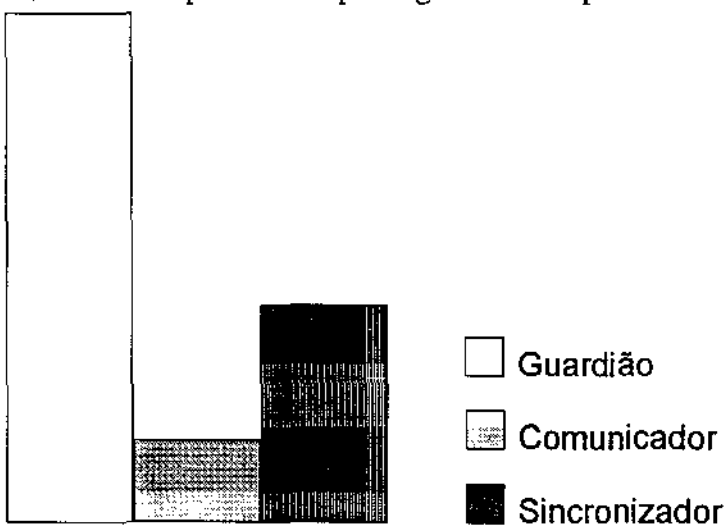


Figura 11 - ForComment.

Existe claramente uma oportunidade de melhoria da qualidade dos sistemas através de um maior equilíbrio destes componentes fundamentais, que exploraremos em nossa própria proposta de modelo, a ser apresentada no capítulo 6.

3.4. AÇÕES AD-HOC E ESPECIFICAÇÕES

Na maioria dos sistemas de workflow existentes, toda ação é normalmente vista como sendo necessariamente executada sob domínio da especificação. Em outras palavras, existe pouco ou nenhum espaço para que os agentes realizem ações não previstas nos modelos dos processos. O problema desta abordagem está relacionado ao tratamento das exceções, como observado por Barthelmess e Wainer em [BW95a]: a não ser em processos extremamente bem comportados, e ocasionalmente mesmo nestes, ocorrem situações não antecipadas que não se adequam à seqüência especificada. Como os processos estão imersos em uma realidade externa que foge ao controle dos sistemas, o atraso na chegada de mercadorias, a perda de algum documento, informações incompletas e outras situações tão comuns no dia-a-dia das organizações fazem com que ações adicionais corretivas precisem necessariamente ser empregadas.

As ações realizadas pelos agentes podem ser divididas, portanto, em dois grupos:

- Ações sob controle das especificações de processo - são as ações antecipadas, ou "normais";
- Ações *ad-hoc* - Ações não previstas nas especificações que precisem ser realizadas;

Note que a "normalidade" de uma ação é determinada unicamente pelo fato de estar ou não prevista na especificação. Isto pressupõe que os modelos de processo representam fielmente a realidade, o que sabemos que não corresponde à verdade.

É fácil constatar que falhas de especificação e situações particulares não contempladas nas especificações terminarão fatalmente por gerar uma exceção. Compartilhamos a noção expressa por Ellis em [EN93a p. 13] de que a alta freqüência de ocorrência de fenômenos não previstos faz com que o termo exceção não seja adequado, por normalmente significar algo raro.

Distinguiremos no presente trabalho as **exceções tratáveis** ou **contingências**, aquelas que podem ser corrigidas através de funcionalidade *ad-hoc* oferecida pelo sistema, das **exceções de sistema**, que fogem totalmente do escopo de funcionalidade oferecido, mesmo de forma *ad-hoc*. O objetivo de um sistema é, portanto, o de minimizar o volume de exceções de sistema, provendo tratamento a nível de especificação ou *ad-hoc* (fig. 12).

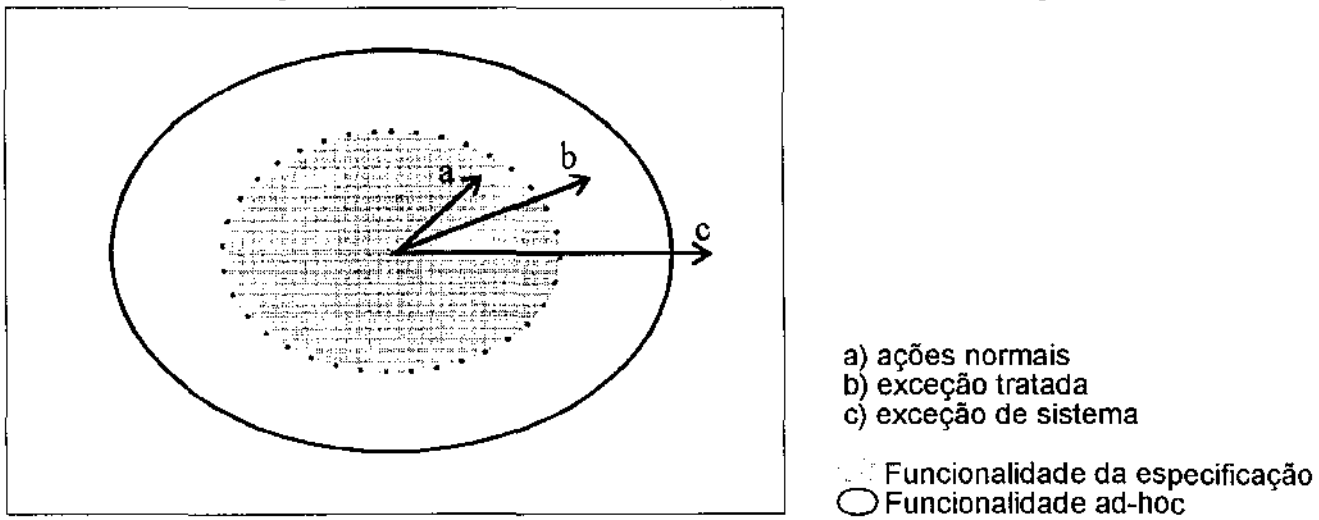


Figura 12 - Fronteiras de especificação e *ad-hoc*.

O grau de utilidade (U) de um sistema é evidentemente relacionado com o volume de ações comportadas por cada uma das fronteiras da figura 12, a de especificação e de ações *ad-hoc*, respectivamente.

A ampliação destas fronteiras está sujeita a limitações. A primeira limitação diz respeito à ignorância dos projetistas em relação aos fatores envolvidos no trabalho ("o modo como as pessoas trabalham é um dos segredos mais bem guardados da América"[10] [Suc95 p. 56]), o que torna inevitáveis as omissões. A segunda limitação é a relativa à complexidade. Modelos mais abrangentes são mais complexos, o que torna a sua

[10] *"How people work is one of the best kept secrets in America".*

construção e manutenção mais onerosa, sendo incompatível com a necessidade já apontada de se permitir a fácil adaptação e evolução [EW94a, EKR95, Sin92].

A figura 13-a apresenta um workflow que oferece poucos recursos, resultando em grande volume de exceções; 13-b equilibra exceções em relação à adaptabilidade; 13-c representa uma solução baseada em linguagem de programação convencional, com grande abrangência de especificação e praticamente zero espaço para ações *ad-hoc*. Existe um espaço por definição não ocupável, independente do poder da especificação, correspondente a exceções de infra-estrutura [SM89, Saa94, BW95a], que dizem respeito à indisponibilidade de recursos, como falhas de equipamento, por exemplo. Por mais amplas que sejam as fronteiras, este tipo de exceção está necessariamente fora do escopo do sistema.

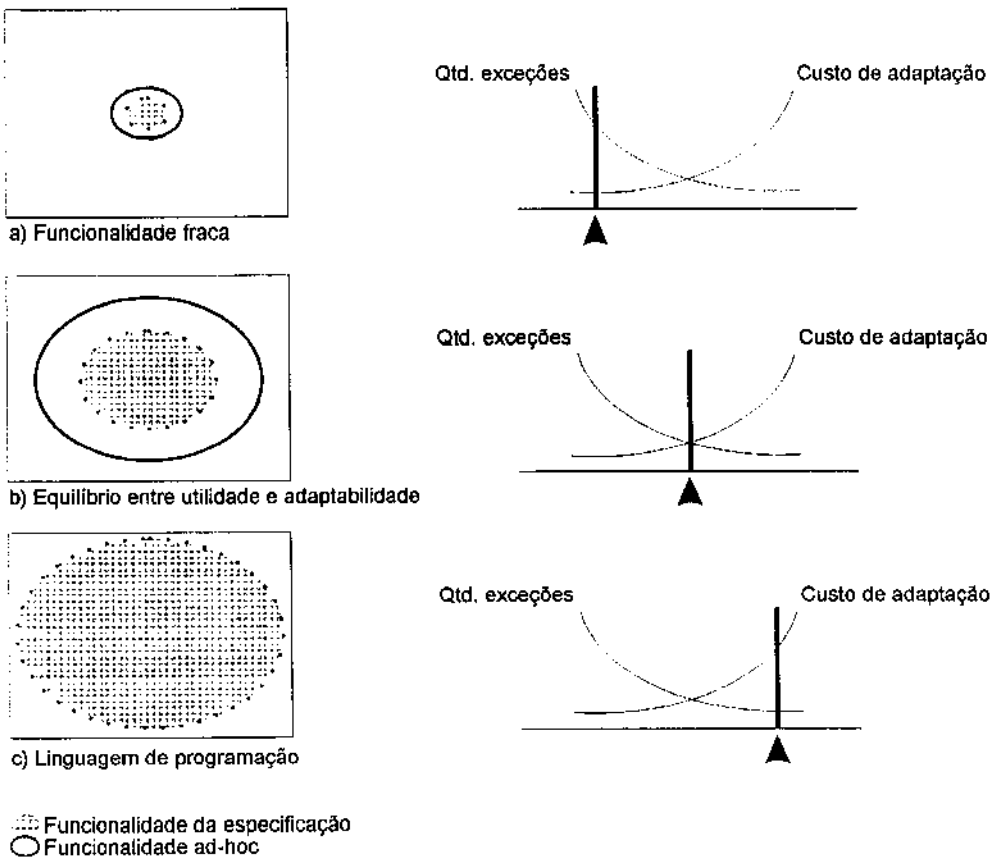


Figura 13 - Volume de exceções *versus* custo de adaptação em diversos sistemas.

A relação entre o volume de ações *ad-hoc* oferecidas e a especificação também é governada por critérios contraditórios. A figura 14 ilustra os extremos desta relação: 14-a apresenta um sistema onde poucas ações *ad-hoc* são permitidas, como em sistemas de workflow atuais, onde as ações são centradas eminentemente no que é possível se especificar. 14-b apresenta o perfil de ferramentas que oferecem apenas suporte a ações *ad-hoc* voltadas à resolução de problemas específicos isolados, com pouca ou nenhuma possibilidade de construção de especificações, como editores de textos ou planilhas, por exemplo.

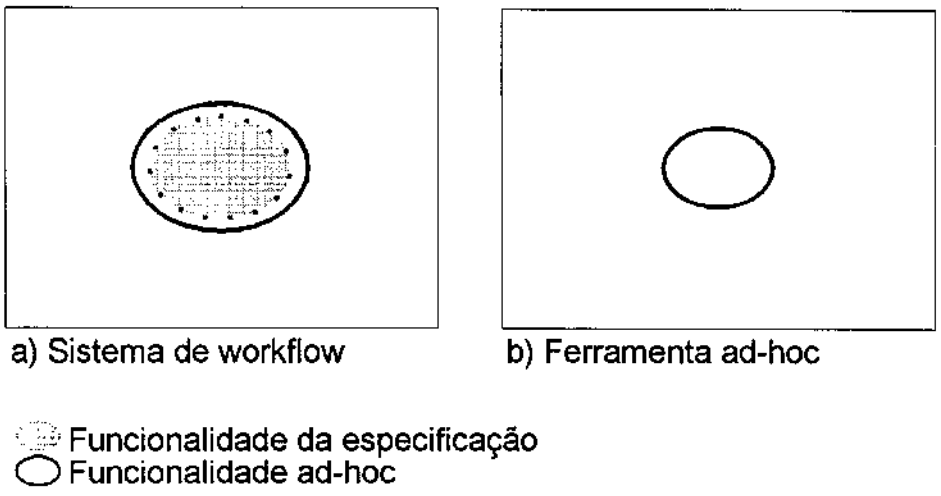


Figura 14 - Ações *ad-hoc* x especificação em sistemas de workflow e ferramentas *ad-hoc*.

A abordagem radicalmente voltada à oferta de ações *ad-hoc* propicia um alto grau de flexibilidade, e o uso não antecipado [Rob93], enquanto que as especificações tornam mais baratas a reexecução de ações repetitivas. Especificações apresentam um custo alto de construção e um custo baixo de reutilização, o diametralmente oposto das ações *ad-hoc*, que podem ser utilizadas de imediato, com pouca ou nenhuma preparação prévia, mas que com a repetição acabam se tornando mais onerosas. O que constitui ação comum repetitiva é uma função do tipo de processo, e provavelmente também sofre evolução no decorrer do tempo, sendo adaptada dinamicamente.

Note que as categorias que enfatizam a ação *ad-hoc*, como os editores e planilhas, por exemplo, são os mesmos que enfatizam o lado guardião, conforme discutimos em 3.3.- ELEMENTOS FUNDAMENTAIS. Sistemas de workflow, por outro lado, enfatizam a especificação, apresentando predominância no aspecto sincronizador. Este alinhamento entre as categorias de sistemas reflete a vocação de cada um dos aspectos: o guardião se presta a suporte a ações *ad-hoc*, enquanto que o sincronizador está naturalmente associado às especificações.

Novamente, existe uma oportunidade de se prover um aumento tanto da funcionalidade na fronteira da especificação, quanto na da ação *ad-hoc*, minimizando-se o número de exceções, através de uma solução que combine os dois componentes de forma harmoniosa.

3.5. VISÕES DE MUNDO

Hirschman alerta em [Hir86] sobre a influência das visões de mundo nas pesquisas relacionadas à Automação de Escritórios, uma área de pesquisa antecessora da de Workflows e que muito a influencia. Neste texto, é apresentada uma divisão das visões de mundo conforme a sua filiação filosófica, sendo afirmado que estas determinam a *priori* o enfoque dado à pesquisa.

Encontramos eco desta posição em [Sac95], onde se afirma que "suposições subjacentes, baseadas em concepções diferentes sobre a natureza do trabalho, coexistem nas

organizações e representam visões de acordo com as quais as pessoas se agrupam"[11] (p. 36). Sachs apresenta como visões antagônicas a baseada em processos ("*process oriented*") e a baseada em atividades ("*activity oriented*").

Blair e Rodden [BR94] associam esta divisão às diferenças de mentalidade de duas comunidades, a de engenharia e de ciências humanas, cada qual com abordagens diferenciadas, principalmente no que tange ao uso de abstrações.

Existem basicamente duas visões antagônicas, que nos remetem à questão ontológica, bem mais antiga, do Realismo x Nominalismo e epistemológica do Positivismo x Anti-positivismo [Bar96]. Estas visões se combinam para formar respectivamente o paradigma Funcionalista (realismo+positivismo) e Interpretivista (nominalismo+anti-positivismo). De forma grosseira, podemos dizer que o Funcionalista acredita no conhecimento absoluto, obtido através da aplicação do método científico, enquanto que o Intepretivista considera o conhecimento como uma conquista comunitária localizada no tempo e espaço. O funcionalista valoriza a abstração, enquanto que o interpretivista valoriza as instâncias. Esta discussão extrapola o domínio da computação, podendo-se encontrar ecos, por exemplo, na área de comunicação, na discussão relativa à objetividade jornalística [BF95].

Percebemos a pertinência desta discussão ao observarmos aqueles textos da área de CSCW onde são discutidos os limites da modelagem na abstração efetiva dos processos de trabalho ([RB91], p.ex.). Estas visões antagônicas se refletem nas opções quanto ao trabalho, aos elementos fundamentais e escolha entre funcionalidade *ad-hoc* ou especificada, como apresenta a tabela da figura 15.

Funcionalismo	Interpretivismo
Modelo	Instâncias
Trabalho individual	Trabalho em grupo
Sincronismo	Guarda de artefatos e Comunicação
Especificações	Ação *ad-hoc*
Divisões em etapas	Atividade comunitária única
Humanos como causadores de erros	Humanos como solucionadores de problemas

Figura 15 - Divisão quanto ao trabalho, fundamentos e ação.

Utilizaremos os termos **interpretivista** para nos referirmos de uma forma sucinta à visão de mundo que privilegia os dados, as ações *ad-hoc*, e o trabalho situado, e **funcionalista** para nos referirmos à corrente complementar, que privilegia o sincronismo, as especificações e a importância das abstrações.

[11] *"Underlying assumptions rooted in different conceptions of work coexist within an organization and represent different lenses through wich people in the organization peer".*

3.6. CONCLUSÕES

Examinamos a posição de um grupo de autores ligados à etnografia, cuja ênfase é o estudo do modo pelo qual o trabalho é efetivamente realizado, obtendo uma série de termos que sintetizam muitas das questões relativas ao trabalho, que teremos ocasião de empregar no restante do texto.

Os elementos fundamentais de sistemas cooperativos são identificados com os aspectos guardião, responsável pelos dados, comunicador, que se preocupa com a interação entre agentes e sincronizador, associado à divisão em etapas e seu disparo em momentos apropriados. Vimos que os produtos apresentam uma predominância intensa em relação a um dos aspectos, resultando em soluções menos equilibradas. Com isto, identificamos uma oportunidade de melhoria dos sistemas, através da oferta de um maior equilíbrio.

Mostramos em 3.4 que a funcionalidade dos sistemas pode ser associada a duas categorias abrangentes, a das ações executadas de forma *ad-hoc* e as executadas sob o comando de uma especificação, e sustentamos que um sistema precisa oferecer suporte relacionado a estes dois modos de uso, o que geralmente não acontece. Novamente, os sistemas se dividem no uso radical de uma forma ou outra de uso, com sistemas de workflow privilegiando as ações sob comando de especificações, enquanto que outras categorias, como editores, planilhas, vídeo-conferência e outros oferecem normalmente apenas ações *ad-hoc*.

Em 3.5.-VISÕES DE MUNDO unimos todas as posições, mostrando que a escolha radical pelo lado guardião ou sincronizador, pela ação *ad-hoc* ou especificação pode ser explicada por um alinhamento de acordo com uma linha filosófica, cujos reflexos não se limitam ao assunto discutido, podendo ser sentidas em outras áreas da ciência. Desta discussão obtemos dois termos, interpretivista e funcionalista, que nos permitem descrever de forma sucinta cada uma das posições antagônicas detectadas, e que procuraremos conciliar em nosso próprio modelo.

4. MECANISMOS BÁSICOS

4.1. INTRODUÇÃO

Examinamos no capítulo passado a questão da construção de um sistema de workflow sob o ponto de vista mais abstrato dos elementos fundamentais e categorias de ações permitidas, identificando duas grandes linhas de pensamento, o interpretivismo e o funcionalismo. Associamos os sistemas de workflow atuais ao funcionalismo, onde predomina a modelagem e o sincronismo em detrimento dos demais aspectos.

Passamos agora a descrever alguns mecanismos básicos que definem o funcionamento de um sistema de workflow propriamente dito, em um nível mais detalhado. Sempre que possível, apontaremos as diversas alternativas de implementação mencionadas na literatura.

A contribuição desta análise está em reunir em um só lugar discussões coletadas a partir de muitas fontes diferentes, geralmente relacionadas a descrições de sistemas. A partir destas discussões poderemos optar pelos mecanismos mais adequados ao modelo que proporemos.

4.2. REDES DE PETRI

Apesar de raramente serem utilizadas diretamente como formalismo de representação de processos, redes de Petri [Rei82] e redes de Petri Coloridas [Jen92] formam a base da maioria dos formalismos utilizados, modificadas cosmética ou semanticamente (em ICN [Ell79], Regatta [SMM+94], RIN [Rei92], p.ex.).

De forma genérica, redes de Petri representam um sistema como uma coleção de lugares (*places*), transições, arcos e marcadores (*tokens*). Lugares contém marcadores e funcionam como entradas para transições. Uma transição acontece quando um marcador se move para outro lugar, se existir um arco entre a transição e o lugar. Antes que a transição ocorra, um marcador precisa estar presente em cada um dos lugares aos quais a transição é conectada por um arco. Na figura 16, por exemplo, o marcador indica a estação do ano atual, a primavera. Quando ocorrer o início do verão, o marcador transita para a próxima estação e assim por diante.

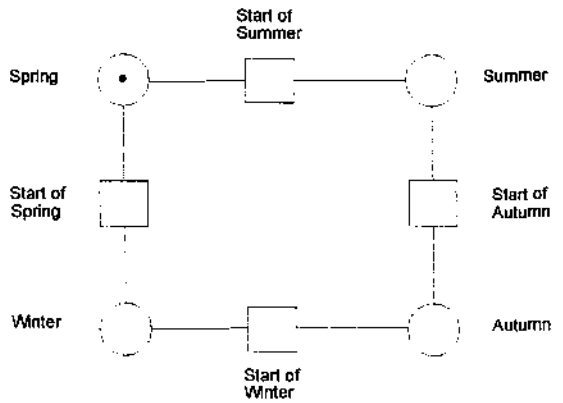


Figura 16 - Estações do ano [Rei82 p. 3].

A possibilidade de uso de múltiplos marcadores para representar a execução concorrente de atividades por diversos agente é um dos atrativos sobre outros formalismos (como máquinas de estado, por exemplo). O estado em redes de Petri é representado pelas posições instantâneas dos marcadores, sendo portanto mais expressivo do que o estado único admitido em máquinas de estado.

4.3. ATIVIDADES

Existe uma infinidade de formas de se representar um processo, como já tivemos ocasião de verificar observando os diagramas apresentados em conjunto com os exemplos da seção 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO. Descartando-se as representações mais exóticas, podemos afirmar que as representações são via de regra gráficas, ou seja, empregam uma linguagem visual, onde as atividades estão representadas por retângulos ou ovais. Ocasionalmente, a descrição do agente (papel) responsável pela sua execução é indicada junto à atividade. Raramente o fluxo de dados é explicitado .

Na maioria das vezes, as atividades representadas graficamente pelos retângulos ou ovais podem corresponder tanto a atividades atômicas, a serem realizadas diretamente pelo agente designado como executor, quanto podem corresponder a atividades abstratas, que por sua vez admitirão subdivisões em etapas distintas, em um segundo nível mais detalhado. Este processo de sub-divisão pode ser repetido por sua nos próximos níveis, normalmente um número não definido de vezes. Como resultado, temos uma árvore de atividades, que tem raiz em uma atividade abstrata, e cujas folhas correspondem a atividades atômicas (fig. 17).

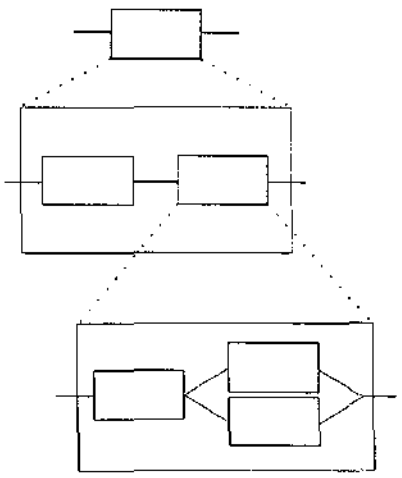


Figura 17 - Hierarquia de atividades.

É feita normalmente a distinção entre atividade manual, realizada por um humano, e a automática, realizada por um componente de software. Os dois tipos de atividades costumam conviver nas especificações.

Quanto à semântica específica de cada atividade, ou seja, quanto as tarefas que devem ser executadas quando cada atividade é efetivamente realizada, o suporte oferecido pelos sistemas costuma variar. Em alguns sistemas (ActionWorkflow, p.ex.), a execução da atividade em si é considerada como algo fora do escopo do sistema. Os agentes são notifica-

dos da necessidade de execução da tarefa e precisam descobrir por seus próprios meios o que deve ser efetivamente realizado. Uma parcela dos sistemas (InConcert [AS94], Regatta [SMM+94], p.ex.) associa a execução da tarefa a formulários e documentos que precisam ser manipulados, tornando a tarefa dos executores mais simples, por já lhes apresentar as ferramentas e informações necessárias. É importante que se note, porém, que mesmo em sistemas que apresentam um lado guardião mais evoluído, a definição dos formulários e outros artefatos manipulados não faz normalmente parte do sistema de workflow em si, i.é, o tratamento dos dados faz uso de ferramentas externas não integradas diretamente.

Alguns sistemas (Regatta [SMM+94] e wOrld [TKF95], p.ex.) admitem a utilização de *scripts*, que são ativados automaticamente no início e final de execução de atividades, o que permite um grau adicional de automatismo associado à execução de cada atividade.

4.4. CONEXÕES E SINCRONISMO

A grande ênfase das representações é no sincronismo entre as atividades. Os retângulos ou ovais são conectados uns aos outros por arestas, e determinam a semântica de sincronismo. Existem dois tipos básicos de possibilidade de conexão entre atividades, a seqüência e o paralelismo. Atividades são necessariamente executadas em seqüência se houver dependência entre elas, podendo ser executadas em paralelo caso contrário. Em outras palavras, se for preciso que a atividade A termine, antes que a atividade B possa ter início, A e B estarão conectadas em seqüência (fig. 18-a). Caso não haja a dependência, A e B podem estar em paralelo (fig. 18-b).

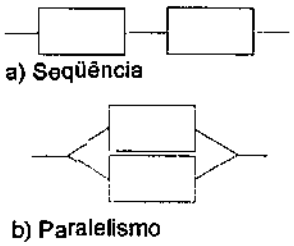


Figura 18 - Conexões entre atividades. a) Seqüência b) Paralelismo.

Além da seqüência e paralelismo, normalmente se pode determinar que algumas atividades são opcionais, ou que apenas algumas das atividades sucessoras possíveis são disparadas. A figura 19 apresenta uma situação deste tipo: a partir de uma atividade de avaliação, apenas um das atividades sucessoras, aprova ou rejeita, será disparada.

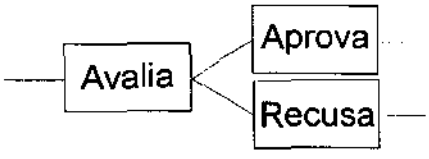


Figura 19 - Atividades alternativas.

A explicitação da lógica de sincronismo desejada é feita normalmente através de conectores de divisão e junção de fluxo. Em termos das redes de Petri, a divisão multiplica o número de marcadores, enquanto que a junção faz o oposto: dois ou mais marcadores são

transformados em um único. A divisão corresponde, portanto, à ativação paralela ou condicional, enquanto que a junção sincroniza diversas etapas anteriores de que uma próxima atividade é dependente.

A maioria dos sistemas reconhece dois tipos de *splits* (separadores), o *and-split* e o *or-split* e dois tipos de *joins* (junção): *and-join* e *or-join*, conforme o glossário da *Workflow Management Coalition* [WMC95], um órgão que busca estabelecer padrões em sistemas de workflow. Um *and-split* ocasiona a ativação simultânea de todas as atividades subseqüentes. *Or-splits* permitem que uma ou mais alternativas sejam ativadas: existe uma escolha entre percursos futuros alternativos. Quanto às operações de junção, no *and-join*, a atividade subseqüente só terá inicio quando todas as atividades precedentes tiverem terminado, significando a dependência em relação a um conjunto de atividades. O *or-join* ocasiona a ativação da atividade subseqüente quando qualquer uma das atividades precedentes finalizar.

A proposta da *Workflow Management Coalition* corresponde ao uso da maioria esmagadora dos sistemas de workflow existentes. Uma proposta superior, com semântica mais poderosa, é feita por Casati et al. em [CCP+95a].

São propostos conectores mais abrangentes: o *fork*, que corresponde ao *split*, e o *join*. *Forks* podem ser **totais**, correspondendo ao *and-split*, quando todos os sucessores são disparados simultaneamente; **condicionais**, correspondendo a um *or-split*; **condicional com exclusão mútua**, que é semelhante ao anterior, mas admite apenas uma única condição verdadeira; **não-determinísticos** - *k* sucessores são escolhidos de forma não determinística e tem sua execução disparada. De novo na proposta podemos identificar a diferenciação entre o condicional e o condicional com exclusão mútua, normalmente não tornada distinta e o não-determinístico, de utilidade questionável.

Quanto aos *joins*, são classificados por [CCP+95a] em: **totais**, correspondentes ao *and-join*; **parciais** - o sucessor é disparado quando *k* predecessores tiverem terminado; **iterativos** - semelhante ao anterior, em que um valor *k* é associado, mas ocorre um novo disparo a cada vez que o quorum mínimo for alcançado, repetidas vezes. Os *joins* apresentam a real contribuição da proposta, pois permitem, através do *join* parcial e iterativo a especificação de uma semântica normalmente não expressável de acordo com a proposta de [WMC95]. Esta semântica pode ser aplicada, por exemplo, no processo examinado em 2.6.2.-REVISÃO DE ARTIGOS, onde o número necessário de laudos de revisão não precisa ser necessariamente igual ao número de revisores, i.é, de cinco revisões podemos aceitar como suficientes as três primeiras. Já o *join* iterativo nos permite disparar a próxima atividade a cada *k* términos de atividades anteriores, o que, como veremos no capítulo 6, nos auxiliará no tratamento de certas atividades batch [BW95a].

4.5. DISPARO DE ATIVIDADES

Vimos que os conectores expressam uma semântica de sincronismo baseada na terminação de eventos anteriores, geralmente o término de uma atividade precedente. Examinaremos agora as diversas opções quanto aos mecanismos de disparo em si:

1) O mecanismo básico está ligado às *t-actions* [EW94b], ações que declaram o término de uma atividade precedente. *T-actions* são os eventos que comumente acionam a mudança de estado, correspondente à movimentação de marcadores em uma rede de Petri.

Este evento é interceptado pelo mecanismo de sincronismo, que interpreta a especificação e emite *s-actions* (*start actions*) que ocasionam o disparo das transições e a conseqüente distribuição de atividades para execução.

As *t-actions* estão normalmente associadas a elementos de interface, como itens de menu ou botões, que são utilizados pelos agentes para declarar o fim do trabalho associado a uma atividade, comandando, conseqüentemente, o prosseguimento do fluxo.

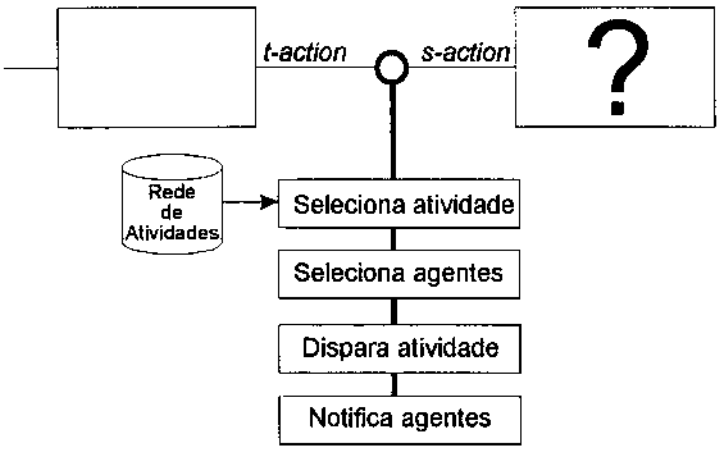


Figura 20 - Mecanismo básico de sincronismo.

2) Alguns sistemas (EACM/EACT [HKE92], $Trigs_{flow}$ [KLR+95], p.ex.) oferecem um mecanismo mais abrangente, baseado em regras, o que permite, além das *t-actions*, a definição de condições de disparo baseadas em condicionais sobre o estado dos dados (fig. 21). O mecanismo de sincronização passa a corresponder ao de um interpretador de regras ECA. Às custas de um aumento de complexidade da linguagem de especificação, ganha-se um poder de especificação que permite inclusive o tratamento de eventos assíncronos, um dos nossos objetivos básicos de aumento de expressividade das especificações.

Concordamos com Casati et al., quando afirmam em [CCP+95b] que regras ECA oferecem um bom mecanismo de explicitação da semântica operacional, e até de implementação do mecanismo de sincronismo, mas não são adequadas para uso direto pelos agentes, pela sua complexidade. Em nosso modelo, a ser apresentado no capítulo 6, utilizaremos uma linguagem visual que procura permitir a flexibilidade e o poder de regras (restritas) através de um formalismo gráfico simples.

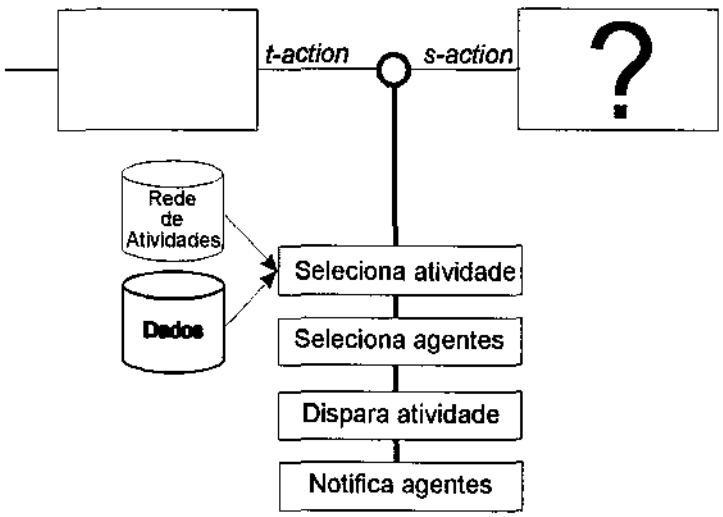


Figura 21 - Influência dos dados na seleção de atividades.

4.6. SELEÇÃO DE EXECUTORES

A questão da seleção do agente apropriado para a execução de cada atividade parece ser considerada secundária na maioria dos sistemas, cujo foco principal é o sincronismo, a modelagem das etapas de processos e suas conexões. Podemos distinguir duas categorias de seleção de agentes:

- Escolha subjetiva manual - corresponde na prática a uma atividade em que o objetivo é o de escolher o executor apropriado para uma ou mais etapas subseqüentes. Em 2.6.- EXEMPLOS DE SITUAÇÕES DE TRABALHO, encontramos duas atividades deste tipo: a atividade de distribuição para avaliação, do processo de seleção de candidatos à pós-graduação (2.6.3), e a atividade, bastante similar, de escolha dos revisores no processo de revisão de artigos (2.6.2).

  Apesar de fundamental, este mecanismo não é explícito na maioria das descrições encontradas na literatura.

- Escolha automática pelo próprio sistema de workflow - nesta modalidade, o próprio sistema se encarrega de selecionar um grupo de candidatos, a partir de descrições, aplicando alguma estratégia de desempate no caso de existirem mais candidatos do que o necessário. As descrições usualmente assumem a forma de um **papel**, que descreve um conjunto de habilidades necessárias para a execução da tarefa, por exemplo, "almoxarife", "secretário", "gerente", e que é associada aos agentes reais através de um relacionamento armazenado no modelo organizacional do sistema. Este esquema pressupõe um modelo simples da organização, onde agentes e papéis são relacionados entre si com cardinalidade m:n. Cada agente pode representar mais de um papel simultaneamente e o mesmo papel pode ser representado por mais de um agente.

A determinação do executor se dá logo depois que a atividade é determinada pelo mecanismo de sincronismo. Informações sobre a atividade, as descrições de papel e o modelo organizacional são utilizadas para determinar o executor (fig. 22).

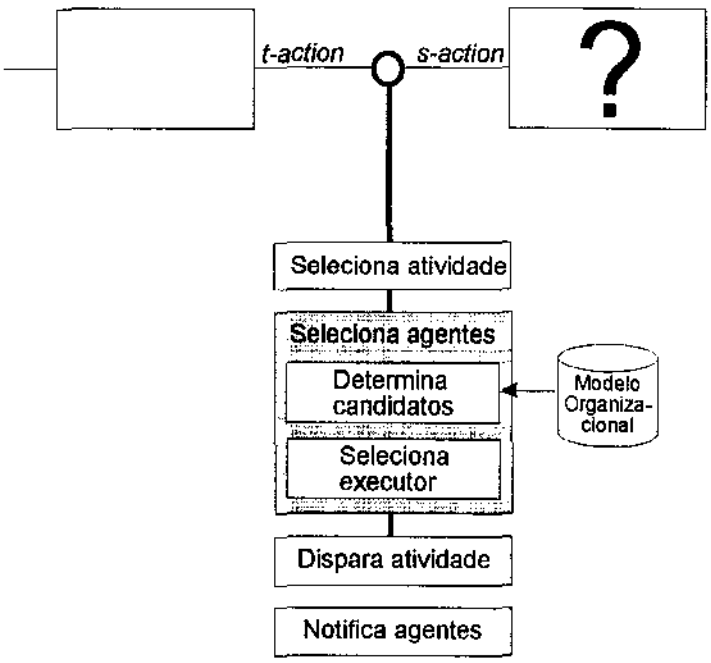


Figura 22 - Determinação de executor.

Vamos examinar os diversos tipos de descrição utilizados:

1) **Papéis simples** - Cada atividade especifica os seus potenciais executores através de um papel (*role*). Este papel simples descreve normalmente uma posição hierárquica, como "gerente", sem que se mencione as unidades organizacionais às quais este papel está associado.

O problema deste mecanismo está relacionado ao fato de que o conhecimento necessário à realização das atividades não é possuído por qualquer agente que atenda à descrição genérica de "gerente", por exemplo. Cada gerente é responsável apenas pelas tarefas relacionadas à sua área de responsabilidade: o gerente financeiro não pode tomar decisões em matérias relacionadas à área de marketing, por exemplo. Em outras palavras, a escolha do agente adequado deve ser situada [BC88] em relação ao contexto da tarefa.

2) **Papéis descritivos** - Um segundo mecanismo de papéis visa contornar o problema apontado na primeira opção, através da ampliação das descrições de papel, que passam a expressá-los em relação aos domínios organizacionais, como por exemplo, "gerente do iniciador do caso", "secretária/o do chefe da equipe x". O papel é baseado em uma relação entre agentes e unidades organizacionais que descreve de forma mais precisa os responsáveis.

A especificação de papéis baseados em descrições relativas às divisões organizacionais pressupõe que exista um modelo mais completo da organização. Encontramos na literatura diversas propostas ([Pri93], [HKE92], p.ex.) de uma modelagem flexível, permitindo a especificação de qualquer informação útil sobre a organização e seus inter-relacionamentos, de forma a representar a estrutura específica da organização.

Mesmo estas descrições mais completas de papéis podem não ser suficientes para a determinação do agente correto em uma situação mais complexa. Examinaremos este aspecto a seguir.

3) **Regras** - Bussler apresenta em [Bus94] um workflow de reembolso de despesa em que a correta determinação do executor da atividade de Aprovação depende de valor específico do caso: se o valor a ser reembolsado for menor do que 5000, ela deve ser realizada pelo gerente da área do iniciador, senão, pelo vice-presidente da área. O mecanismo proposto por Bussler para a solução deste problema envolve o uso de regras que, a partir do exame dos dados específicos do caso, atribuem a atividade a um ou outro papel (fig. 23).

A vantagem da abordagem proposta por Bussler está na possibilidade de se utilizar os próprios dados de um caso como parâmetro para decisões mais sofisticadas, embutidos normalmente em condicionais. Sistemas que não apresentam esta possibilidade deverão tratar esta atribuição de maneira excepcional, envolvendo intervenção humana, mesmo no caso em que a regra de atribuição é conhecida.

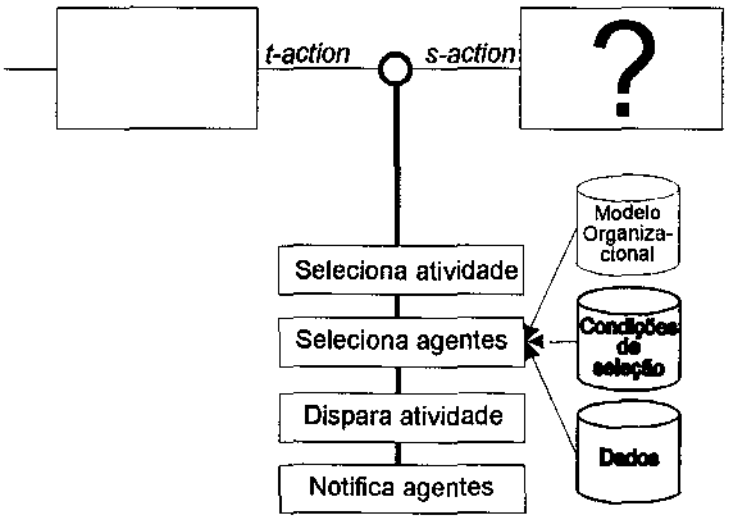


Figura 23 - Influência dos dados na seleção de agentes.

Vamos passar a discutir o que ocorre se a descrição utilizada corresponder a mais de um candidato. Se existirem, por exemplo, cinco atendentes de balcão disponíveis para a realização de um preenchimento de formulário, algum critério de desempate precisa ser utilizado para selecionar o que efetivamente será o responsável pela atividade. O critério mais comum é o da oferta-e-aceite, na qual todos os possíveis candidatos recebem a solicitação simultaneamente e o agente que iniciar a execução por primeiro fica responsável por ela, ocasionando a retirada da solicitação feita aos demais candidatos.

É fácil perceber que este é um problema de alocação de recursos que pode utilizar outros tipos de estratégias, que otimizem esta alocação segundo algum critério. Alguns artigos ([CCP+95a], [KLR+95], p.ex.) mencionam esta possibilidade, sem contudo existirem propostas mais concretas.

Barthelmess e Wainer mencionam em [BW95a, BW95b] que diversos outros algoritmos podem ser empregados (*round-robin*, distribuição por carga de trabalho, p.ex.) ou que a seleção adequada pode envolver julgamento subjetivo, a ser realizado por um gerente, por exemplo.

O pressuposto básico de seleção de agentes para execução das atividades, mesmo quando estratégias mais sofisticadas são utilizadas, é o de que cada atividade será executada por

um único agente. Em teoria, esta limitação não é grave, já que o trabalho coletivo pode ser distribuído de tal maneira que cada atividade corresponda apenas à porção individual de trabalho que cada agente teria de qualquer forma que realizar.

Este pressuposto, de que as atividades são sempre individuais, induz à construção de especificações que em alguns casos não são apropriadas. O isolamento dos agentes em atividades separadas desincentiva (ou até inviabiliza) a sinergia algumas vezes necessária para a execução de alguma atividade. Imagine uma atividade de reunião, por exemplo, em que cada um dos participantes recebe parte das atribuições, executando-as em atividades diferentes e estanques. Mesmo se admitindo que o resultado da soma das ações individuais seja equivalente ao coletivo, o que é questionável, a limitação da comunicação imposta pelo sistema irá provocar pelo menos uma demora acentuada na obtenção deste resultado.

4.8. SISTEMAS BASEADOS EM ATOS DA FALA

Uma parcela importante de sistemas é baseada em teorias de linguagem e comunicação, como as propostas por Searle. Estes sistemas (Coordinator [Win88], Domino [KHK+91], ActionWorkflow [Med92], p. ex.) tornam explícitas as intenções de cada mensagem trocada, baseados em um critério de classificação expresso em uma da teoria de atos de fala (*"speech act"*), proposta por Searle em [Sea69, Sea79] ou em variantes.

A base desta teoria é de que a unidade mínima de comunicação não é a sentença ou outra expressão, mas sim promessas ou solicitações que refletem a tentativa de fazer com que o ouvinte execute algo [DW91] (a fala como causadora da ação). Searle categoriza os tipos de atos de linguagem, de forma que um conjunto finito deles seria em princípio capaz de descrever qualquer situação. Winograd e Flores contribuem criando uma estrutura que aglutina estes atos isolados, admitindo que cada um deles tenha apenas um conjunto limitado de respostas (fig. 24). Esta malha básica é a unidade a partir da qual redes de sequenciamento de trocas de mensagens são formadas.

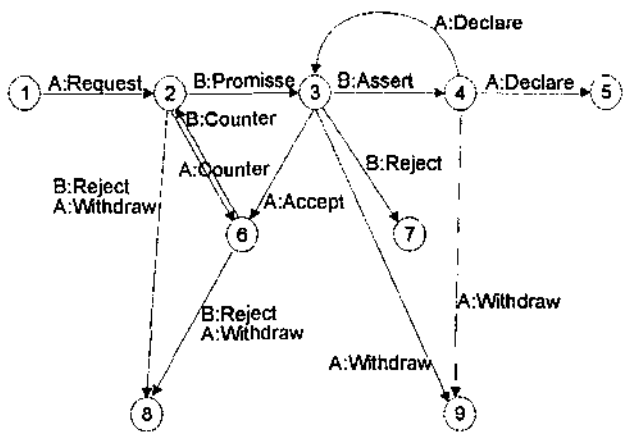


Figura 24 - Malha básica de conversação [BC88 p.491].

A aplicabilidade desta teoria em sistemas de workflow decorre diretamente do fato de que a linguagem, sendo geradora de ação, pode ser utilizadas para modelar os processos organizacionais, através do registro da seqüência de trocas envolvidas.

Sistemas construídos de acordo com esta perspectiva (The Coordinator [Win88], ActionWorkflow [Med92], Domino [KHK+91], p.ex.) parecem resultar surpreendentemente rígidos, apesar de se basearem em algo tão fluido quanto a linguagem.

O que parece ocorrer nestes sistemas é uma excessiva ritualização da comunicação, estabelecendo-se padrões rígidos que parecem ser incompatíveis com as seqüências reais de troca utilizadas em escritórios [Suc93]. A semelhança entre as redes de sequenciamento e as utilizadas em sistemas sincronizadores é tamanha, que se pode realizar a tradução de especificações construídas utilizando-se esta teoria para outras especificações que utilizam o formalismo de redes de Petri, a base dos sistemas sincronizadores (veja por exemplo [EM93]). Na prática, o que parece acontecer é que os usuários simplesmente ignoram as classificações dos tipos de mensagens, utilizando o sistema como se fosse um correio eletrônico [Suc93].

ActionWorkflow [Med92] propõe a representação através da composição de diagramas mais simplificados, onde são expressas apenas as quatro fases básicas de **solicitação**, **concordância**, **execução** e **aceite**, envolvendo sempre um cliente e um executor (fig. 25).

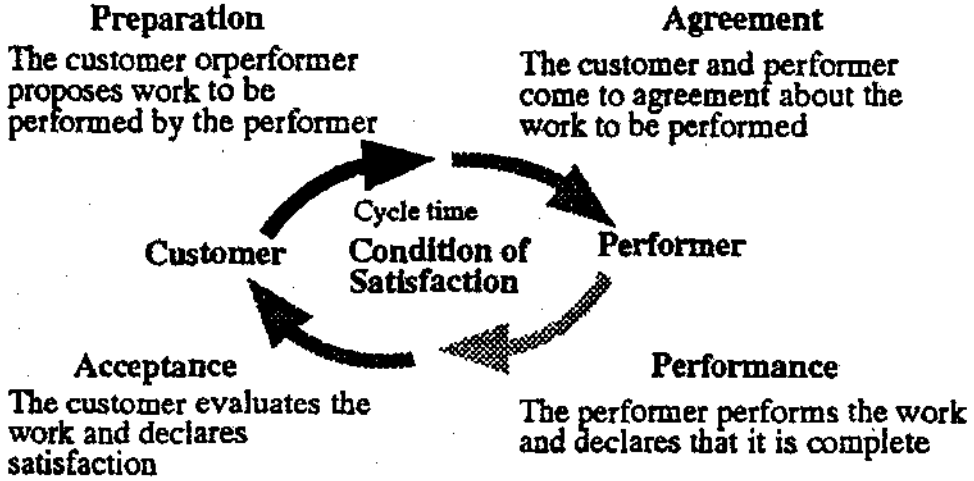


Figura 25 - Cadeia de workflows em ActionWorkflow [MWF93 p.49].

Cada uma das quatro fases oferece possibilidades limitadas de interação, i.é, as mensagens permitidas em cada uma destas etapas de negociação são pré-estabelecidas. Na fase de concordância, por exemplo, pode-se fazer **promessas**, **recusar**, ou **contrapropor**; a execução admite apenas a **declaração de término** ou o **cancelamento**, e assim por diante.

Cada um dos compromissos é chamado de workflow. O conjunto dos workflows, possivelmente encadeados, modela o funcionamento de um processo, como pode ser visto na figura 26.

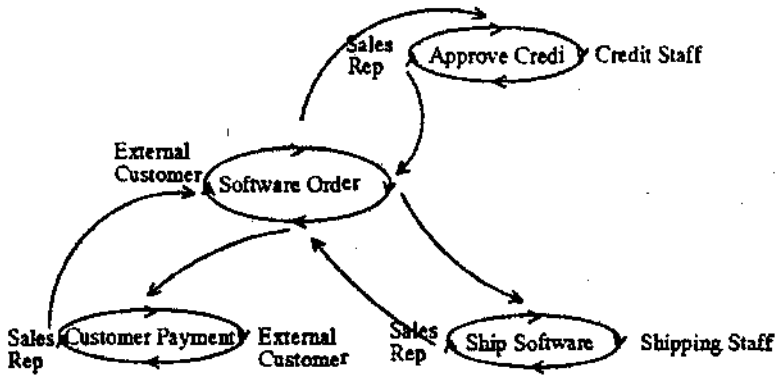


Figura 26 - Composição de workflows [MWF93 p. 50].

Observe no diagrama da figura 26 que a interligação entre os workflows pode ser estabelecida em qualquer uma das quatro etapas principais. Cada uma destas ligações significa que esta fase corresponde na verdade a um sub-workflow. A fase de **concordância** do workflow principal (Encomenda de Software), por exemplo, é realizada através do sub-workflow de Aprovação de Crédito. O mesmo acontece na fase de **execução** e **aceite**, que iniciam a execução de Expedição e Pagamento, respectivamente.

O poder deste tipo de especificação está na flexibilidade de negociação que oferece. Para aquelas situações em que determinadas atividades são feitas e refeitas, em um laço de controle de qualidade, até que atinjam um patamar de satisfação determinado, a representação utilizada parece ser bastante sucinta e efetiva. É questionável, porém, a quantidade de processos que necessitarão deste tipo de funcionalidade. De qualquer forma, através dos conectores, como o *fork* e o *join*, examinados em 4.4.-CONEXÕES, é possível se estabelecer estes ciclos através de arestas que ligam atividades a suas predecessoras, obtendo-se uma funcionalidade equivalente sempre que necessário (fig. 27).

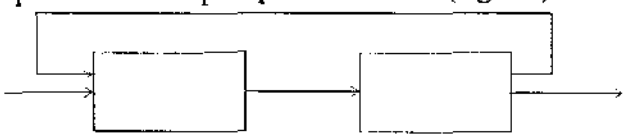


Figura 27 - Ciclos de qualidade utilizando-se conexões entre atividades.

4.9. CONCLUSÕES

Vimos que as redes de Petri oferecem um formalismo básico a partir do qual mecanismos específicos de sincronismo podem ser derivados. Os marcadores, sua movimentação e o disparo que ocasionam oferecem um vocabulário útil na descrição dos mecanismos, realizada a partir da seção 4.3.

Examinamos algumas formas de representação de processos, normalmente especificados através de ovais ou retângulos, representando atividades, ligados por arestas através de conectores que determinam a semântica de sincronismo oferecida por cada sistema. Discutimos o disparo de atividades em termos de eventos, principalmente as *t-actions* e *s-actions*, eventos de terminação e disparo de atividades, respectivamente. Mostramos

também a vantagem de se permitir que os dados dos casos possam influenciar a tramitação, através de expressões condicionais associadas a disparo de atividades e seleção de executores, evitando que um tratamento excepcional, manual, tenha que ser empregado.

Mostramos que a seleção de agentes, apesar de relegada normalmente a um segundo plano, apresenta problemas interessantes, relacionados à utilização de descrições mais poderosas de agentes, de seleção condicional, baseada em dados, e do uso de estratégias alternativas de alocação, como *round-robin*, carga de trabalho e assim por diante.

Finalmente, em 4.6, discutimos brevemente a corrente ligada à teoria de atos de fala, que apesar de seus eventuais defeitos constitui ainda uma facção importante na área de workflows. Sustentamos que o modelo mais genérico de conectores pode ser utilizado em substituição do mecanismo associado a atos de fala, úteis apenas em situações específicas, onde ciclos de qualidade precisam ser estabelecidos.

5. REQUISITOS DESEJÁVEIS

5.1. INTRODUÇÃO

Examinamos nos capítulos 3 e 4 os fundamentos do problema e os mecanismos básicos utilizados em sistemas de workflow, respectivamente. Vamos agora estabelecer os requisitos desejáveis, a partir dos conceitos já enunciados de maior equilíbrio entre componentes e semântica de especificação mais poderosa.

A análise da literatura da área nos permitirá verificar como as mesmas questões são encaradas por outros autores e quais são as alternativas envolvidas. Nossa discussão envolverá principalmente os sistemas Regatta [SMM+94, SIM94, SI95], InConcert [AS94, Inc96a, Inc96b], EuroCoOp EACM/EACT [HKE92] e wOrlds [FTK95, TKF95, BK95].

Consideramos que todos estes sistemas aderem, em maior ou menor grau, de forma intencional ou não, a uma linha comum identificada com o equilíbrio de fatores que julgamos necessário e que iremos explorar em nossa própria proposta. Nossa ênfase será nos aspectos que julgamos importantes e que não são cobertos de forma adequada, segundo nossa opinião, em sistemas existentes.

5.2. O SISTEMA "IDEAL"

Identificamos no capítulo 3 os elementos fundamentais em sistemas colaborativos, associados a três aspectos, o guardião, o comunicador e o sincronizador. Vimos ainda que as ações podem ser divididas em ações *ad-hoc* e as executadas sob controle de especificações.

Chamamos a atenção para o fato de que os sistemas atuais apresentam uma polarização em relação a estes aspectos, podendo ser divididos em dois grandes grupos, que associamos ao interpretivismo e ao funcionalismo. Os sistemas de tendência interpretivista seriam aqueles que possuem aspectos guardião e comunicador fortes, com pouco ou nenhum sincronismo, pressupondo o uso de ações *ad-hoc*. Os funcionalistas, pelo contrário, priorizam o aspecto sincronizador em detrimento dos outros dois aspectos, enfatizando a especificação.

Ao observarmos os exemplos apresentados em 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO, podemos notar que atividades individuais e coletivas se entremeiam em um mesmo processo, como, por exemplo, no processo de seleção de candidatos à pós-graduação, em que a atividade de avaliação, como vimos, pode tanto ser executada de forma individual, com cada agente avaliando os candidatos de forma isolada, como pode ser conduzida de forma coletiva, com diversos (ou todos) professores reunidos para realização do trabalho de forma coletiva. A limitação imposta pela maioria dos sistemas atuais, capazes de apoiar apenas atividades individuais, nos parece arbitrária. Em termos do gráfico de influência dos aspectos, que apresentamos em 3.3.-ELEMENTOS FUNDAMENTAIS, desejamos que os três aspectos básicos apresentem equilíbrio, como exibido na figura 28.

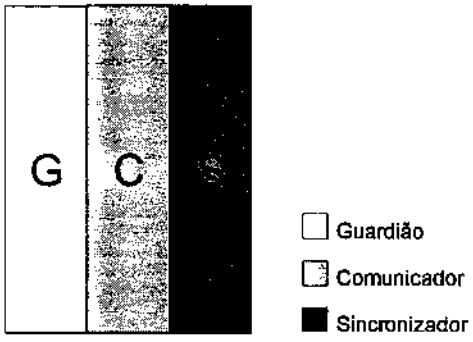


Figura 28 - Equilíbrio dos elementos fundamentais.

Obviamente, este equilíbrio não implica na necessidade de que todo processo utilize estes três componentes de forma absolutamente equilibrada. Dependendo das características de cada processo, estes podem, e provavelmente apresentarão, uma predominância localizada de um dos componentes. Em outras palavras, em processo em que a comunicação for fundamental, predominará o aspecto comunicador, da mesma forma que em outros a guarda de artefatos ou o sincronismo podem predominar devido a características do objetivo de negócio que se deseja incorporar.

Quanto ao suporte a ações *ad-hoc* e especificações, associados ao tratamento de situações excepcionais e rotineiras, respectivamente, também podemos estabelecer critérios desejáveis de cobertura e equilíbrio (fig. 29).

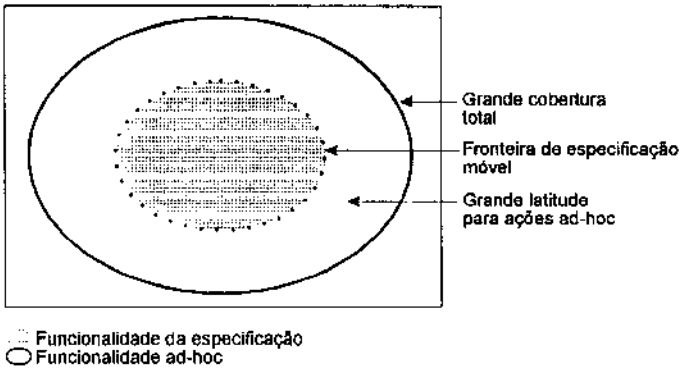


Figura 29 - Funcionalidade *ad-hoc* e de especificação.

Desejamos ampla cobertura do universo de possibilidades, de forma a minimizar o número de contingências que se transformam em exceções de sistema: queremos aumentar a cobertura do sistema, mantendo um grau aceitável de adaptabilidade;

Dentro do possível, desejamos que as ações especificáveis possam também ser realizadas de forma *ad-hoc* e vice-versa: que qualquer ação *ad-hoc* possa ser incluída em uma especificação, i.é, que não haja diferença entre estes dois níveis de funcionalidade.

São duas as conseqüência desta igualdade: por um lado, não se obriga o usuário a utilizar a especificação apenas para fazer uso de uma função não disponível de forma *ad-hoc*. Por outro lado, qualquer ação pode ser incluída na especificação, ou seja, não existe nenhuma ação que precise necessariamente ser feita de maneira *ad-hoc* por não serem especificáveis.

A fronteira de especificação deve poder ser movida dinamicamente, tanto para ampliá-la em tempo de execução, incluindo-se novas funções, como para diminuí-la, substituindo-se ações programadas por outras *ad-hoc*.

Consideramos que não só a separação entre **executor e planejador** é prejudicial, como também o é a própria separação entre **execução e planejamento**. Em outras palavras, não só as mesmas pessoas devem realizar tanto as tarefas de execução como as de planejamento, mas além disto não deve existir um modo especial de planejamento, que deve se confundir, dentro do possível, com a própria execução. Esta discussão nos remete à questão da separação entre modelagem e execução, que parece não ser apropriada em sistemas de workflow. O paradigma mais apropriado seria nestes sistemas o das planilhas (conforme [SMM+94] e [LMY88], p.ex.), onde execução e especificação são absolutamente homogêneas.

A figura 30 apresenta a combinação dos dois gráficos das figuras 28 e 29. A oferta de recursos associados aos três aspectos básicos, o guardião, o comunicador e o sincronizador, apresentam influência semelhante, podendo ser utilizados tanto de forma *ad-hoc* quanto incluídos em uma especificação.

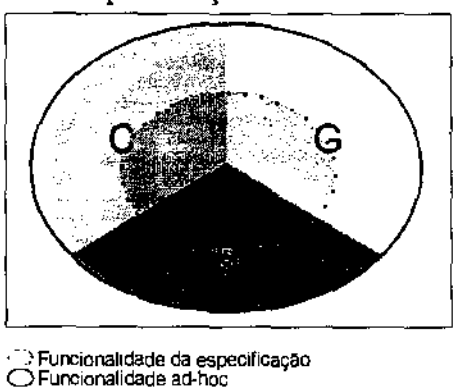


Figura 30 - Aspectos fundamentais combinados a ações *ad-hoc* e especificações.

Estabelecidas as linhas gerais do que consideramos desejável, vamos examinar mais detalhadamente os requisitos que decorrem desta posição de equilíbrio desejada, apresentando as opções relacionadas na literatura, quando existirem.

5.3. INTEGRAÇÃO DO PLANEJAMENTO E EXECUÇÃO

Esta questão tem a ver com a já comentada mudança de paradigma tornada necessária pelo aspecto dinâmico dos casos a serem tratados por sistemas de workflow. Conforme discutimos em 2.5.-O DESAFIO EM SISTEMAS DE WORKFLOW, a multiplicidade de tipos de organizações, processos, modos individuais de se atingir objetivos, casos especiais e evolução da realidade externa agem como fatores que inviabilizam a divisão tradicional entre modelagem, executada por um grupo de técnicos, e execução, a cargo dos usuários dos sistemas de workflow. Em sistemas de workflow, esta fronteira entre modelagem e execução precisa ser tornada muito mais tênue por vários motivos:

1) Swenson et al. apontam em [SIM+94, SMM+94] a necessidade de que os planos utilizados em sistemas de workflow sejam desenvolvidos com o máximo envolvimento possí-

vel daqueles que efetivamente executam o trabalho, e não por um grupo de especialistas que estabelecem de cima para baixo o que deve ser feito e como. Esta posição é coerente com as observações de diversos autores ligados à etnografia ([Suc83], [RB91], [Sac95], p.ex.), que revelam as incorreções decorrentes de descrições feitas por quem não executa o trabalho;

2) A idéia de que existe um modo único de se atingir determinado objetivo parece não encontrar respaldo na observação do trabalho real [Hir86, Saa94]. Cada organização, unidade organizacional e agente pode (e normalmente tem) seu modo particular de conduzir seu trabalho;

3) Quando surge um caso especial, um pedido de urgência, por exemplo, não se pode esperar que o pessoal técnico seja acionado, execute uma fase morosa de análise e programação, para que então o caso possa ter prosseguimento. Problemas precisam ser resolvidos à medida em que surgem, pelos próprios usuários. Os mesmos argumentos se aplicam, em certa medida, a situações em que os processos evoluem por força de mudança na realidade externa;

4) Em algumas situações não rotineiras, para as quais não existe um processo definido, como no processo que tem início com o pedido da presidência de que seja criado um novo produto (descrito em 2.6.5.-CRIAÇÃO DE UM NOVO PRODUTO), o planejamento constitui provavelmente a parte mais importante do trabalho, e é feito normalmente à medida que o caso evolui, e não de forma antecipada;

As regras funcionam nos escritórios como recomendações de caráter geral, existindo inclusive regras contraditórias. A aplicação inteligente destas regras, feita pelos agentes, é que permite que o trabalho seja feito [Suc87].

5.4. AÇÕES *AD-HOC* E ESPECIFICAÇÕES

Por melhor que seja o mecanismo de especificação de um sistema, ele sempre será um empecilho para os usuários, constituindo um trabalho gasto em ações que não se traduzem diretamente em resultados, um gasto burocrático feito em nome do bom funcionamento do sistema. A especificação não é um objetivo por si, mas um modo de se armazenar seqüências repetitivas de ações, de forma a diminuir seu custo de repetição, caso sejam corriqueiras.

As atividades devem ser estruturadas de tal forma que permitam um grande grau de liberdade de realização de ações, sem que seja necessário se recorrer ao planejamento a todo o momento [EN93a]. A maior liberdade de ação a nível de tarefa garante, segundo Galbraith, a diminuição do número de exceções [Gal77]. Existindo maior liberdade de ação no âmbito da própria atividade, é natural que menos problemas tenham que ser resolvidos fora dele.

Partimos do pressuposto de que não deve haver impedância entre as ações passíveis de execução *ad-hoc*, e aquelas incorporadas em especificações. Em outras palavras, toda a ação, na medida do possível, deve poder ser executada tanto por comando direto do usuário, quanto constar de uma especificação.

As ações podem ser agrupadas de acordo com algumas categorias:

- Básicas - a mais básica das ações é a de disparo de atividades. Esta ação deve dar início à seleção dos agentes que serão responsáveis pela realização das tarefas, avisando-os de que seus serviços são solicitados e dando a eles o acesso aos dados correspondentes. Além desta ação, podemos ter ações de notificação, que servem para avisar os agentes de algum fato importante, como a aproximação de uma data limite, por exemplo;
- Controle de execução - certas situações exigem que se possa suspender, cancelar ou encerrar atividades em execução, ou delegar/reatribuir uma atividade para outros agentes. Estas ações estão normalmente associadas a tratamento de situações excepcionais, como a falta de um agente, por exemplo.
- Ações sobre dados - são as ações que permitem a criação e manipulação de instâncias de objetos e documentos relativos a um caso. Ações deste tipo costumam ser especialmente ausentes na maioria dos sistemas.

As ações básicas normalmente serão embutidas em especificações, enquanto que as demais serão provavelmente efetuadas de forma *ad-hoc*. A possibilidade, porém, de se utilizar qualquer ação nos dois modos permitirá maior flexibilidade no uso do sistema.

5.5. MAIOR PODER DE ESPECIFICAÇÃO

Fora as ações, que devem poder ser executadas de forma *ad-hoc* ou embutidas em especificações, como já discutimos em 5.4.-AÇÕES AD-HOC E ESPECIFICAÇÕES, é preciso que os mecanismos de sincronismo, exclusivos das especificações, possam corresponder à semântica necessária para se expressar as dependências reais dos processos.

Estes elementos de sincronismo determinam basicamente quais ações devem ser disparadas a cada momento, e quais os agentes mais apropriados para sua execução. Se feitas de forma *ad-hoc*, os próprios usuários determinam a seqüência de ações, o momento de disparo e os executores apropriados, baseados em seu conhecimento do problema. Na especificação, o julgamento dos agentes é substituído por elementos que registram esta lógica de disparo.

A semântica de sincronismo da linguagem de especificação deve, portanto, ser suficientemente poderosa para que se possa expressar por meio dela as condições de disparo usuais dos processos. O exame das situações de trabalho, relatadas em 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO, nos permite verificar que existem situações comuns que não podem ser especificadas pela maioria dos sistemas de workflow atuais, como passaremos a examinar.

5.5.1. EVENTOS ASSÍNCRONOS

Muitos eventos são gerados fora dos limites das organizações, não podendo existir um controle por parte da organização quanto ao momento de chegada de um deles. Solicitações a entidades externas, como a compra de componentes, por exemplo, serão atendidas de uma forma assíncrona, i.é, não existem garantias precisas quanto ao momento real de sua chegada. A maioria das especificações não permite a explicitação destes eventos esperados de forma clara.

Além disto, é flagrantemente ausente na grande maioria dos sistemas a possibilidade de representação do tratamento de eventos assíncronos ocasionais [BW95a, BW95b]. Estes são eventos que podem ou não ocorrer, dependendo de fatores externos, como um pedido de cancelamento: não se pode saber a *priori* se haverá ou não um cancelamento. A abordagem empregada nos sistemas examinados se restringe a tratar este tipo de evento como exceção, independentemente da freqüência com que este ocorre e da existência ou não de uma rotina para o seu tratamento. Em outras palavras, mesmo que o cancelamento seja freqüente, não existe modo de incluí-lo na especificação, sendo obrigatório o seu tratamento de forma *ad-hoc*.

Casati et al., em uma rara referência a este assunto, propõem em [CCP+95] um modelo onde cada tarefa pode especificar um conjunto de pares <Exceção, Reação>, que nada mais são do que regras de tratamento de eventos assíncronos opcionais. A exceção é um predicado que pode incluir aspectos temporais e relacionados a estados de objetos. A reação é escolhida a partir de um conjunto restrito de opções que incluem a terminação da tarefa, o seu cancelamento e a emissão de notificação. Este mecanismo pode ser utilizado, por exemplo, para emitir notificação (ou terminar a tarefa) automaticamente, quando a data limite para sua execução for ultrapassada (op. cit. p.4).

Diversos eventos das organizações são baseados em tempo. Atividades normalmente tem prazos e processos ou atividades de processos podem ser iniciados automaticamente em determinados pontos do tempo. Em nosso entender, este tipo de evento pode ser tratado através de um mesmo mecanismo genérico de tratamento de eventos assíncronos. Não é o que acontece na maioria dos sistemas:

1) O sistema Regatta oferece *timer-nodes* que podem ser programados para desativar (!) após um determinado tempo, que pode variar de minutos a meses após a ativação. A desativação dispara ações de terminação que correspondem ao efeito desejado, por exemplo, a emissão de notificação. O propósito deste mecanismo é, segundo Swenson et al., o de "permitir que eventos do tipo lembrete sejam gerados após o transcurso de um tempo determinado" [SMM+94 p.24].

2) InConcert e ActionWorkflow permitem a especificação de regras ativadas com base no tempo (veja a discussão em 4.10.4.-SCRITPS E AÇÕES AUXILIARES). O mecanismo de regras deste sistema é tratado de forma independente das especificações em si, não sendo claro o modo pelo qual são criadas e ativadas.

5.5.2. ATIVIDADES BATCH

A não ser pelos textos referentes à nossa própria pesquisa ([BW95a] e [BW95b]), as atividades batch, apesar de bastante comuns, não são mencionadas por outros autores de que tenhamos conhecimento. Atividades batch são aquelas atividades em que um conjunto de casos precisa ser reunido para sofrer algum tratamento que pressupõe a comparação relativa entre eles, geralmente associado a otimizações.

Um exemplo de atividade deste tipo pode ser encontrado na seleção de candidatos à pós-graduação, na atividade de Classificação final (veja 2.6.3.-SELEÇÃO DE CANDIDATOS À PÓS-GRADUAÇÃO). Na Classificação final, a decisão de aceitar ou não cada um dos candidatos é tomada através da comparação relativa entre todos eles, a partir das avaliações realizadas na fase anterior pelos professores revisores. Esta classificação só pode ser

realizada se os dados de todos os casos estiverem disponíveis simultaneamente. Pode-se considerar que o objetivo da classificação final é o de otimizar o preenchimento de vagas, que devem ser ocupadas pelos melhores candidatos.

A existência de atividades batch parece decorrer justamente da necessidade de se aplicar alguma otimização que exija a comparação entre casos. Em uma companhia telefônica, por exemplo, os pedidos individuais de ligação feitos pelos usuários podem ser reunidos para que se possa agrupá-los de acordo com o local onde deve ser feita a instalação. Desta forma, os instaladores não precisam perder tanto tempo se locomovendo entre os locais de instalação, já que estes estarão previamente agrupados por proximidade.

Outros exemplos incluem a ordenação para estabelecimento de prioridades de atendimento, com solicitações de urgência e de clientes preferenciais sendo executadas antes das demais, ou o agrupamento de pedidos de forma a se otimizar a utilização de equipamento, como no agrupamento por cor, tipo ou volume.

O que existe de especial na atividade batch é o fato de que precisamos especificar uma sincronização **entre casos diferentes**, e não entre atividades do mesmo caso, como é comum. A sincronização se dá em uma outra dimensão, a do processo como um todo (todos os candidatos do processo de seleção, p.ex.).

Como os sistemas não oferecem mecanismos que possibilitem a especificação deste tipo de sincronismo, problemas deste tipo acabam tendo que ser tratados pelos próprios agentes, através de algum artifício.

5.5.3. ATIVAÇÕES PARALELAS

Em algumas ocasiões, mais de uma instância da mesma atividade precisa ser executada em paralelo, como por exemplo, em uma atividade de revisão, em que desejamos que vários revisores executem a mesma atividade simultaneamente.

Caso se possa determinar a *priori* qual o número de instâncias simultâneas desejadas, a representação deste tipo de atividade é bastante simples, bastando que se explicite cada uma das instâncias no modelo, como feito, por exemplo, em InConcert [MS93]. Neste caso, qualquer variação no número de instâncias desejadas é tratada como exceção, envolvendo a intervenção de um agente.

Existem casos, porém, em que a quantidade de instâncias só pode ser determinada em tempo de execução, como por exemplo, quando queremos executar um número variável de atividades de revisão, de acordo com o tipo do artigo, que pode necessitar um número maior ou menor de revisores dependendo do assunto e complexidade (veja 2.6.2.-REVISÃO DE ARTIGOS).

São raros os sistemas e textos que tratam desta funcionalidade:

1)No sistema Regatta o *split-stage* [SMM+94], permite a replicação baseada na quantidade de agentes disponíveis para a execução da atividade no momento do disparo, i.é, se apenas dois revisores estiverem disponíveis, duas atividades paralelas serão criadas; se forem quatro os revisores, são criadas quatro atividades paralelas.

2) Casati et al. propõem em [CCP+95a] a *multitask* como mecanismo genérico de especificação de ativações paralelas. Cada *multitask* especifica um valor *j* qualquer que controla o número de ativações paralelas desejadas. Assim que é disparada, são criadas *j*

instâncias paralelas da atividade especificada. A *multitask* será considerada como encerrada assim que *k* das instâncias tiverem terminado. Podemos, por exemplo, especificar que serão ativadas j=5 atividades paralelas, e que consideraremos este conjunto de tarefas terminado quando as primeiras 3 tiverem terminado. Apesar de nada ser mencionado por [CCP+95a], existe um limite prático para o valor *j* que dita o número de ativações paralelas em uma *multitask*: o número de agentes disponíveis. Dificilmente fará sentido que a mesma atividade seja realizada pelos mesmos executores mais de uma vez. Suponha que criemos quatro atividades de revisão replicadas, atribuindo-as aos dois únicos agentes disponíveis para a sua execução. Cada um deles teria que realizar a mesma atividade duas vezes!

5.5.4. ATIVAÇÕES CONDICIONAIS BASEADAS EM DADOS

Nem todas as atividades constantes de uma especificação serão obrigatoriamente disparadas durante a execução de um caso: certas atividades só devem ser realizadas em determinadas condições. Suponha que o Processamento de Pedidos inclua uma etapa adicional no caso de pedidos muito grandes. Devido ao impacto que ocasionam na produção, estes pedidos podem possuir uma fase extra de análise e programação de produção, ausente nos pedidos menores. O fator que determina a execução da atividade está relacionado aos dados do caso, mais especificamente à informação sobre quantidade do pedido. Esta ativação baseada em dados está representada pelo diagrama da figura 30.

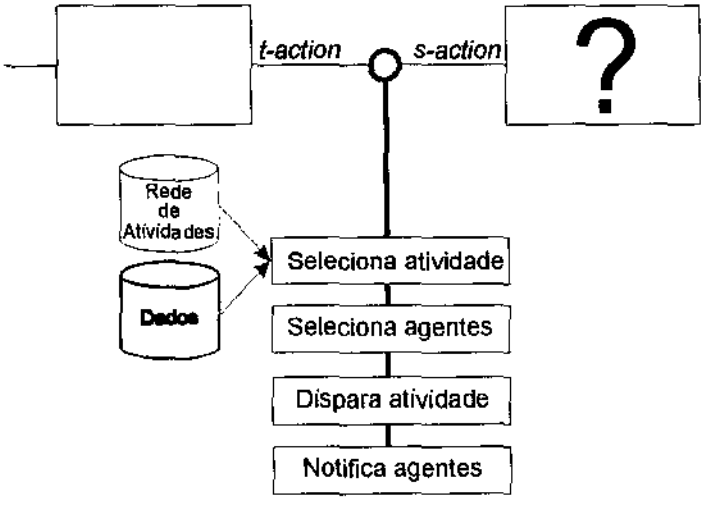


Figura 30 - Ativação baseada em dados.

Kappel et al. apresentam uma variante da ativação condicional, em que o estado de um banco de dados determina a necessidade de ativação de uma atividade [KLR+94]. Durante o processo de aquisição de insumos (veja 2.6.1.-AQUISIÇÃO DE INSUMOS), a atividade de cotação de preços só precisa ser realizada caso as validades das propostas, armazenadas em um banco de dados, já estejam vencidas. Novamente, o que determina ou não a execução da atividade é o estado dos dados.

Na maioria esmagadora dos sistemas, enfatiza-se apenas as decisões baseadas na ação de um agente, que teria neste caso que verificar se as cotações estão ou não atualizadas, de

forma manual, para então dar prosseguimento ao caso, através de ativação de item de menu ou botão.

Apesar de não enfatizarem este aspecto, diversos sistemas oferecem algum suporte a este requisito, de maneira mais ou menos abrangente:

1) O sistema Regatta oferece, de maneira tímida, um mecanismo que examina qualquer atributo em testes de condições às quais estão atrelados eventos de disparo de estágios [SMM+94]. O mecanismo não é explicitado, e sua existência é mencionada em um único parágrafo do texto citado;

2) No sistema wOrlds, a ativação de uma ação pode ser protegida por uma condição guardiã booleana, que só permite o disparo se for verdadeira [TKF95], o que torna possível a explicitação da semântica desejada.

3) EACM/EACT [HKE92] permite a especificação de pré e pós-condições que indicam condições de disparo baseadas em predicados sobre dependências de recursos em geral: atividades, objetos e até recursos externos, como salas de reunião, por exemplo.

4) Em $\text{Trigs}_{\text{flow}}$ [KLR+95], regras ECA são utilizadas para especificação do disparo de transições e diversas outras tarefas do sistema. Um substrato ativo é adicionado ao banco de dados utilizado pelo sistema, permitindo que regras sejam utilizadas em associação a determinadas classes do sistema ou de forma independente.

Dentre os mecanismos citados, regras apresentam o maior potencial em relação a este requisito, por permitirem a especificação direta de condições baseadas em dados.

5.5.5. SELEÇÃO DE EXECUTORES

Além do disparo de ações, e da sincronização, um papel importante dos sistemas de workflow é o da determinação dos agentes que deverão se encarregar da execução de cada atividade. Conforme discutido em 4.6-SELEÇÃO DE EXECUTORES, podemos distinguir duas categorias de seleção de agentes, a manual, realizada por um agente, e a automática, realizada pelo sistema.

No caso da seleção automática, os mecanismos envolvem algum tipo de descrição de classes de agentes habilitados a executar cada uma das tarefas, que associamos a **papéis simples**, **papéis descritivos** e **regras**, cada qual com um crescendo de expressividade. Em tempo de execução, estas descrições são transformadas em referências a agentes reais, que se encarregarão da execução das tarefas.

Quando mais de um candidato corresponde às descrições, uma estratégia de escolha deve ser empregada. Quando existirem, por exemplo, diversos secretários habilitados para a execução de uma única atividade, é preciso decidir qual deles será efetivamente o responsável pela execução. A mais comum é a de se oferecer a tarefa a todos os candidatos e aguardar até que o primeiro deles inicie a sua execução. Basta então retirar a tarefa das pendências dos demais candidatos.

Uma referência rara a estratégias alternativas é feita em [KLR+95]. Neste artigo é mencionada a possibilidade de se empregar outros tipos de estratégia, como alocar a tarefa ao candidato com a menor carga de trabalho, por exemplo.

De forma geral, existe a restrição de que de todos os candidatos apenas um será selecionado. Esta restrição está relacionada com a noção quase unânime de que atividades são

sempre realizadas individualmente, o que consideramos inapropriado, por impedir que atividades eminentemente coletivas sejam representadas e executadas sob o controle do sistema. Existem evidências de que este tipo de atividade é bastante comum nas organizações, principalmente em casos que venham a envolver o tratamento de exceções [SMS94].

A distribuição de tarefas, principalmente quando se utiliza papéis descritivos e regras, pressupõe a existência de um modelo da organização, que descreve as unidades organizacionais, agentes e o relacionamento entre eles. O modo diverso pelo qual as organizações são estruturadas impede que um esquema único possa ser utilizado para representá-las [Pri93], i.é, as variações não acontecem apenas na estensão, mas nas próprias entidades e relacionamentos empregados. Esta característica obriga que a modelagem seja feita através de uma ferramenta que permita a especificação tanto do esquema quanto da estensão. Em outras palavras, devem poder ser especificados os grupos, comitês, departamentos, divisões e o conjunto de agentes que pertencem a cada um destes grupos ou unidades.

1) Em EACM/EACT, dois meta-objetos básicos permitem a representação de entidades e relacionamentos arbitrários. Um *browser* permite a exibição de detalhes de entidades, da rede de entidades e relacionamentos e uma visão hierárquica baseada em relacionamentos específicos, como de departamento, ou de papéis. Este browser permite pesquisa e geração de relatórios, além do uso de marcadores para acesso rápido a informações muito consultadas [HKE92].

2) Em Tosca [Pri93], são oferecidos um servidor e um *browser*. O servidor permite o acesso a informações organizacionais a outros aplicativos, via uma API. Novamente, o modelo permite a definição livre de entidades e relacionamentos e seu refinamento através de herança. O escopo do sistema é bastante abrangente, incluindo informações de agentes, unidades organizacionais e outros recursos da organização, como objetos, regras e especificações.

3) Bussler apresenta em [Bus94] um modelo flexível, que também permite a representação de organizações arbitrárias. O principal atrativo deste sistema é o uso de regras para a incorporação das políticas de seleção de executores, com o uso de condicionais e teste do estado dos dados. Pode-se, por exemplo, determinar que uma atividade de aprovação de crédito deve ser executada por um gerente, caso o valor deste crédito não ultrapasse certo limite, ou por um diretor, no caso de valores superiores.

Da mesma forma que no disparo condicional de atividades baseado em dados (veja 4.5.4.- ATIVAÇÕES CONDICIONAIS BASEADA EM DADOS), o teste do estado dos dados durante a escolha de executores permite o registro de condições que de outra forma exigiriam a atuação de um agente humano (fig. 31).

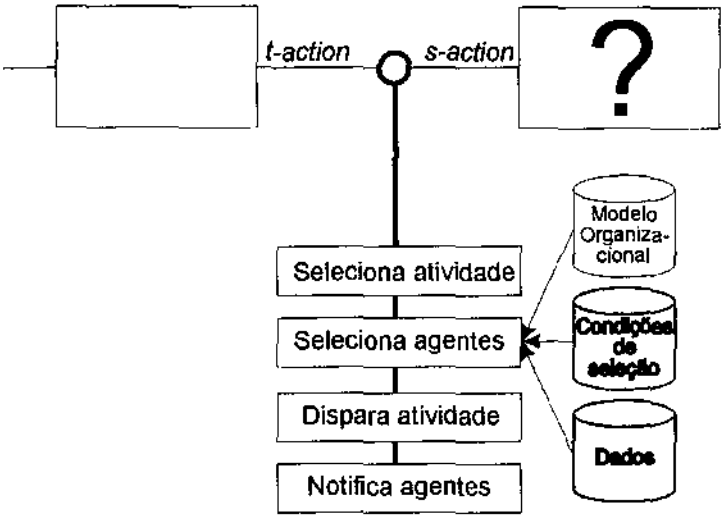


Figura 31 - Seleção de agentes baseada em dados.

5.6. AMBIENTE DE EXECUÇÃO

Definimos como ambiente de execução o conjunto de ferramentas de manipulação disponíveis aos usuários para facilitar a realização do trabalho em si. Em sistemas de workflow, a parte visível deste trabalho se relaciona às manipulações de informações através das quais os agentes registram os dados da realidade que são relevantes em cada caso.

Apesar da importância desta manipulação de informações em qualquer sistema, este aspecto costuma ser relegado a um segundo plano, ficando à sombra da descrição dos aspectos de sincronismo, que é o grande tema da maioria dos artigos. Apesar disto, pode-se extrair das entrelinhas destes textos algumas posições interessantes a respeito deste aspecto, que passamos a discutir.

Uma leitura atenta revela que os ambientes de execução estão normalmente associados a alguma forma de contexto, onde residem as informações manipuladas durante a execução das atividades. É importante ressaltar que o conceito de contexto como o apresentaremos não é mencionado na literatura, constituindo uma contribuição de nosso trabalho.

A característica básica, comum a todas as facetas do contexto é a da possibilidade de modificação dinâmica de seu conteúdo [WB96]. O requisito de adaptação dinâmica dos dados é complementar ao dinamismo relacionado aos processos, conforme discutimos em 2.5.-O DESAFIO EM SISTEMAS DE WORKFLOW. Da mesma forma que as etapas dos processos podem variar de organização a organização, grupo a grupo, ou em casos especiais, também os dados apresentam características diferentes em momentos diferentes. Em outras palavras, da mesma forma que não se pode antecipar com precisão as etapas de um processo, também não se pode antecipar com precisão as necessidades de dados, que sofrem as mesmas influências das especificidades de caso e dos agentes que executam as tarefas associadas a eles.

Podemos distinguir duas dimensões complementares relativas aos contextos: 1) a do seu tempo previsto de duração e 2) a do número de agentes que atua sobre o contexto simultaneamente.

1) Quanto ao tempo previsto de duração, os contextos podem assumir diversas facetas complementares, como ***workspace***, ***folder*** e **ambiente virtual**:

- **Contexto como *workspace*** - o contexto funciona como espaço de trabalho, reunindo as informações, objetos e ferramentas necessárias à execução de uma tarefa ou atividade específica. O tempo de duração do *workspace* normalmente é o da duração da atividade a que ele diz respeito: ele é criado no início da execução da atividade e eliminado ao seu término.

  A contribuição desta faceta é a de que a execução em si é contemplada através da oferta das ferramentas para a sua realização, de uma forma integrada. Tudo o que é necessário para a realização de uma tarefa passa a estar facilmente disponível para os agentes.

- **Contexto como *folder*** - segundo esta faceta, o contexto corresponde a um conjunto de documentos e objetos que pode ser encarado como uma pasta que contém as informações relacionadas a um determinado caso. Assim como em uma pasta comum, das utilizadas em escritórios, esta composição pode ser modificada a qualquer tempo, através da adição/remoção de objetos e ferramentas. Esta visão é apresentada por diversos autores ([Kar90], [EB82], p.ex.), geralmente ligada ao conceito da circulação de pastas (*circulating folders*). Nesta faceta, a duração esperada normalmente é a do próprio caso: ao criarmos um novo caso, este é associado a uma nova pasta, que será utilizada até que o caso se encerre, quando então pode ser eliminada (ou armazenada em um arquivo morto ou similar).

  A principal contribuição relacionada a esta faceta é a de permitir que os casos possam ser referenciados como se possuíssem a existência física de uma pasta comum de escritório.

- **Contexto como ambiente virtual** - esta faceta está ligada a contextos mais duradouros, que reúnem os artefatos necessários à execução de tarefas recorrentes. Um ambiente virtual pode ser comparado a uma sala aparelhada para a realização de uma tarefa específica, como uma reunião, por exemplo. O escopo de duração de contextos deste tipo costuma ser mais permanente, independendo de casos e processos específicos. Diversos casos diferentes, eventualmente correspondentes a processos também diferentes podem ser tratados em contextos deste tipo. Estes contextos normalmente fazem mais sentido quando utilizados de forma colaborativa, ou seja, envolvendo a participação simultânea de diversos agentes, que se reúnem neste "ambiente" para solucionarem algum tipo de problema. A questão do individual e do coletivo nos leva à nossa segunda dimensão, que discutiremos a seguir.

2) Além do aspecto da duração que acabamos de discutir, qualquer uma das facetas de contexto expostas acima pode ser utilizada de forma **individual**, por um único agente que trabalha em isolamento, ou pode permitir a participação **coletiva** de diversos agentes, que operam, através da mediação do contexto, de forma síncrona ou assíncrona, para resolverem em conjunto algum problema.

- **Contexto de suporte ao indivíduo** - esta é a utilização dominante, utilizada na maioria dos sistemas de workflow existentes, que pressupõem justamente que as tarefas coletivas são divididas de tal forma que cada agente receba atribuições individuais e

realize estas tarefas de forma estanque, com nenhuma ou pouca comunicação com os demais agentes. O problema desta visão é que ela é incompatível com muitas das atividades que acontecem nas organizações, especialmente no caso de contingências. Uma pesquisa realizada por Saastamoinen et al. [SMS94] identifica que apenas 3,3% das exceções são tratadas a nível individual. As demais (96,7%) envolvem na sua solução um grupo, o escritório ou toda a organização.

Esta característica do tratamento de exceções é provavelmente uma das principais razões da inadequação dos sistemas de workflow existentes, que provêem suporte apenas para indivíduos e não para grupos, como necessário. Podemos associar o uso individual do contexto à visão funcionalista, que vê os processos como linhas de montagem, com cada agente agindo como um processador ou engrenagem independente.

- **Espaço compartilhado de participação** - de acordo com esta visão, o contexto é encarado como uma espaço compartilhado onde diversos agentes cooperam para que um determinado objetivo seja alcançado. Possui uma lista de participantes, que ao mesmo tempo recebem direitos especiais de acesso sobre o contexto e passam a ser notificados de eventos significativos associados a ele. Além das mensagens geradas pelo próprio sistema, os participantes podem trocar mensagens entre si livremente, além de poderem atuar sobre os objetos de forma simultânea.

  A noção de contexto como ambiente compartilhado de participação tem um embasamento em teorias ligadas à etnografia (como a de Strauss [Str93], p.ex.). A principal contribuição desta faceta é oferecer um mecanismo que permite a atuação de vários participantes simultaneamente, executando ações de forma mais livre do que é possível sob o domínio de atividades isoladas, correspondendo a uma ampliação simultânea dos aspectos guardião e comunicador, ligado, portanto, à corrente interpretivista.

O ponto principal a ser destacado em relação ao contexto, principalmente se a participação coletiva for permitida, é a maior liberdade propiciada por este mecanismo para a realização de ações *ad-hoc*, que em princípio são impedidas ou limitadas pelo mecanismo tradicional de atividade. As ações agora se realizam no contexto, e não mais sob o controle de um plano que estabelece uma seqüência aceitável de etapas isoladas, existindo uma maior influência dos aspectos guardião e comunicador.

Vamos examinar como os contextos são tratados em alguns sistemas:

1) A forma mais primitiva de contexto, se é que se pode dar este nome ao mecanismo simples proposto pelo sistema InConcert, é a de permitir que cada atividade possa especificar o conjunto de objetos de que necessita [AS94, MS93], correspondendo, portanto, à faceta de *workspace* individual.

2) O sistema Regatta possui o conceito do colóquio ("*colloquy*") como espaço que contém os objetos relacionados a algum processo [SIM+94, SMM+94]. O colóquio não é vinculado a nenhuma atividade, independendo delas, correspondendo, portanto, à faceta de *folder*. A limitação neste sistema é que as ações não podem ser realizadas diretamente no contexto, devendo ser necessariamente associadas a atividades. Os agentes podem enviar mensagens aos outros participantes do colóquio, incluindo texto, voz ou documentos em *attach*, mas apenas associadas a opções escolhidas a partir de atividades, explicando a

razão da escolha, "principalmente se uma opção não padrão for escolhida" [SMM+94 p.21], por exemplo, a rejeição de um pedido de compra. Podem ser criados sub-contextos, mas estes são isolados do contexto mais externo (super-contexto). O sub-contexto funciona como mecanismo de garantia de privacidade para trechos do processo para os quais não se deseja divulgação geral do andamento e dos dados específicos utilizados.

Apesar de existir a noção de múltiplos participantes, que são os agentes que executam atividades relacionadas a um caso de forma concorrente a cada momento, a interação entre estes agentes é mínima e podemos dizer que só é permitido o uso individual do contexto.

3) Em EACM/EACT, os contextos já comportam a troca de mensagens e modificação explícita da composição dos participantes de forma desvinculada das atividades. Cada subtarefa do contexto possui seu próprio sub-contexto. Estes sub-contextos podem compartilhar recursos do super-contexto [HKE92]. Um aspecto especial é de neste sistema os contextos podem ser desvinculados de tarefas ("*unbounded contexts*"), funcionando como artefato que reúne pessoas e objetos necessários à obtenção de algum objetivo para o qual não se conhece ou não se deseja subdivisão em etapas (*brainstorming*, p.ex.). Podemos associar estes usos às facetas de *folder* e ambiente virtual, respectivamente, admitindo por uma vez uso coletivo: enquanto um caso está em execução, o contexto funciona como um *folder* compartilhado pelos diversos participantes. Quando o caso termina e se opta por utilizar o contexto de forma desvinculada de tarefas ("*unbounded*"), a faceta correspondente é a do ambiente virtual compartilhado, onde diversos agentes "se reúnem" para utilizar os artefatos constantes do contexto como subsídio para as suas discussões.

4) No sistema wOrld [FTK95, TKF95, BK95], a idéia do contexto é levada adiante, encontrando um embasamento teórico na **teoria da ação** (*action theory*) de Strauss [Str93] (citado em [FTK95]). O contexto, que recebe o nome de *locale* neste sistema, corresponde à parte passível de implementação em computador do *social world* de Strauss.

Um *locale*, como o nome indica, se baseia em um paradigma espacial. Agentes "entram" e "saem" dos *locales*, como se estes fossem ambientes virtuais onde se reúnem os que estão trabalhando em um determinado assunto. O sistema permite interação síncrona, via Internet, através de vídeo e voz. Mensagens assíncronas podem ser enviadas a caixas-postais ou a secretárias eletrônicas. A visualização de participantes atende ao critério de ciência (veja 5.7.1.-AWARENESS), possibilitando que cada agente saiba quem está trabalhando em tarefas relacionadas a cada momento. Neste sistema, a faceta de ambiente virtual coletivo é a central.

O suporte a atividades coletivas, impõe uma série de requisitos adicionais, que a maioria dos sistemas de workflow preferem ignorar. São estas características que permitem que ocorra a sinergia necessária em trabalhos em grupo, onde o produto do trabalho coletivo é superior à soma das partes individuais. Isto se deve à influência mútua que a ação de cada agente exerce nos demais: o rumo tomado por um dos agentes influencia os demais, que passam a agir de forma diferente do que fariam se estivessem em isolamento. As características necessárias, de ***awareness***, **comunicação** e **planejamento colaborativo**, que passaremos a apresentar em seguida, são mais comuns em sistemas que identificamos com a corrente interpretivista, como editores e planilhas colaborativas, por exemplo.

5.6.1. *AWARENESS*

Um requisito fundamental em atividades coletivas é o referente à *awareness* (ciência). Este termo se refere ao fato de que os participantes precisam estar constantemente cientes (*"aware"*) dos componentes que os cercam e das modificações significativas destes componentes. Como as ações são coletivas, é preciso que todos os envolvidos sejam constantemente alimentados com informações que os posicionem em relação às ações realizadas por outrem [FTK95]. Esta noção compartilhada da ação propicia a sinergia necessária aos grupos, permitindo que cada participante realinhe seus micro-objetivos em função das tendências expressas pelas ações dos demais participantes.

De forma geral, podemos dizer que devem ser difundidas informações sobre modificações na composição do grupo de participantes, modificações de objetos, modificação de planos e realização de atividades.

1) Um dos mecanismos mais explorados é o que informa quais são os participantes trabalhando na mesma tarefa simultaneamente. Em Grove, ícones com retratos dos participantes são exibidos pela interface do editor [EW94b]. Em wOrld, imagens de vídeo dos participantes são compartilhadas. Os agentes podem percorrer os vários *locales* a que tem acesso, verificando quais são os participantes do momento. Este mecanismo é semelhante ao utilizado no sistema The Cruiser [EW94a p. 66], que permite a varredura dos escritórios de outros agentes para estabelecimento de comunicação, com a diferença importante de que em wOrld os ambientes são virtuais.

2) Em EACM/EACT, todas as mensagens de ativação e término de tarefas, bem como qualquer outra mensagem informal produzida por qualquer participante, são enviadas para todos os participantes e observadores de forma automática, além de serem armazenads em um *log* para consulta futura.

3) [FTK95] menciona diversas pesquisas relacionadas ao sistema wOrld para estender os mecanismos de *awareness* para além do existente: acompanhamento da manipulação de objetos compartilhados, permitindo a visualização destas manipulações por qualquer participante; monitoração de eventos externos ao ambiente, como modificações ocorridas no sistema de arquivos (*file system*); mecanismos de *feedback* visual para apresentação de modificações cumulativas, como as realizadas nas especificações, por exemplo (p. 14).

5.6.2. COMUNICAÇÃO

Segundo Ellis e Wainer, uma função principal de groupware é o suporte à comunicação e colaboração entre os participantes [EW94b p. 86]. Apesar disto, na maioria dos sistemas de workflow, a comunicação se restringe à assíncrona, geralmente atrelada à execução de atividades.

A separação entre a comunicação informal e a formal é diagnosticada por Kreifelts et al. como um empecilho ao bom funcionamento do sistema [KHW93]. [KHK+91] relata o sentimento de "aprisionamento" de usuários por dificuldade de comunicação: "funcionários se sentiam 'encurralados' pela funcionalidade do sistema, sentindo falta de

uma transição mais harmoniosa entre atividades de processos e outras formas mais informais de comunicação"[12] [KHK+91 p.127].

1) A necessidade de se prover um mecanismo de anotações *ad-hoc* é destacada em diversos textos ([Pri93], [KHK+91], [SMM+94], [BW95a], p.ex.). Este mecanismo propicia que sejam adicionados comentários livres, de forma dinâmica, a qualquer objeto que se deseje. Prinz associa este uso ao de um quadro-branco (*white-board*), uma aplicação tradicional de groupware que permite o compartilhamento de anotações [Pri93 p.149]. A anotação funciona como mecanismo de comunicação implícita assíncrona entre os agentes.

2) Em Regatta, a cada opção tomada podem ser associadas mensagens sumarizando o que os outros participantes do colóquio precisam saber a respeito das razões que levaram a esta decisão. Estas mensagens podem incluir texto, voz ou outros tipos de documentos [SMM+94 p.21].

3) O sistema EACM/EACT [HKE92] inclui um mecanismo de conferência, similar aos encontrados em BBSs (*Bulletin Board Systems*), onde são armazenadas as mensagens formais e informais, servindo de história acumulada da comunicação referente a cada caso.

4) No sistema wOrld, os *locales* oferecem um paradigma espacial que permite a comunicação direta entre participantes que estejam executando tarefas relacionadas de forma simultânea, simulando um ambiente virtual [FTK95]. A comunicação pode ser realizada de forma síncrona, através de vídeo e voz, ou podem ser deixadas mensagens em caixas postais ou secretarias eletrônicas.

5.6.3. Planejamento colaborativo

Definimos como planejamento colaborativo a construção cooperativa de planos, envolvendo um conjunto de agentes diretamente envolvidos na determinação da seqüência de etapas de trabalho desejada. O fruto do planejamento colaborativo é um meta-objeto que estabelece como outras atividades futuras passarão a ser tratadas: uma especificação.

A possibilidade de planejamento colaborativo não se restringe à construção de novas especificações, mas é tão ou mais importante no replanejamento, durante a execução de um workflow. O agente que detecta uma contingência nem sempre estará capacitado a resolvê-la. Em alguns casos, nenhum dos agentes isoladamente terá condição de replanejar sozinho uma reação, sendo necessário o envolvimento, eventualmente em diversas etapas, de um grupo maior.

1) O sistema Chautaqua [Ell95] possui um modo de planejamento colaborativo chamado *town-meeting*. O *town-meeting* é moldado nas reuniões realizadas em aldeias para decisão comunitária de aspectos de interesse geral. De forma semelhante, diversos agentes se congregam e estabelecem através de discussão qual a estratégia mais apropriada a ser empregada. O produto é uma especificação de sequenciamento de atividades.

2) No sistema wOrlds [FTK95], o sistema é usado de forma reflexiva, i.e., através dos mecanismos usuais de interação do sistema são produzidas especificações. Assim como no *town-meeting*, diversos participantes cooperam simultaneamente (ou não) para a cons-

---

[12] *"Officials felt 'fenced in' by the system features they had to use. A smooth transition from office procedures processing to more informal ways of communication [...] was missed."*

trução da especificação. A vantagem deste sistema sobre o Chautaqua, por exemplo, é que este planejamento não constitui uma ferramenta especial, mas utiliza os recursos disponíveis para qualquer atividade. O que varia é apenas o produto, que é um meta-objeto, uma especificação, e não um objeto de aplicação.

5.6.4. INSPEÇÃO

Uma função importante dos sistemas de workflow é a de fornecer informações de acompanhamento do andamento dos casos. Na maioria dos sistemas, esta tarefa é tratada como função independente das demais, sendo muitas vezes implementada através de ferramentas externas ao sistema em si.

Ellis e Wainer estabelecem em [EW94b] a existência de quatro tipos diferentes de inspeções:

- de participante;
- por caso ("*endeavor based*");
- total;
- de segunda ordem ("*second order*")

1) A forma mais corriqueira de inspeção e que constitui um aspecto fundamental dos sistemas, sem o qual se tornam pouco úteis, é a de **participante**. Este tipo de inspeção permite a cada agente saber quais são as atividades da sua competência a cada momento, servindo geralmente como mecanismo de acesso a estas atividades. Este acaba sendo o componente através do qual o sistema comunica ao agente a necessidade de sua participação em alguma atividade.

A forma mais comum de organização deste componente é a de lista de afazeres (*to-do list*), em que as atividades e eventualmente seus prazos são listados. [HKE92] utiliza um modelo alternativo, baseado em uma lista indentada, na forma de tópicos ("*outline*"). A vantagem desta última é que o contexto de execução de uma atividade, os objetivos de mais alto nível de que faz parte, podem ser facilmente visualizados. [EW94b] destaca que a visualização apresentada deve preferencialmente ser determinada segundo preferências de cada agente (por data, tipo de atividade, etc.).

Na maioria dos sistemas, a inspeção de participante acaba fornecendo o único mecanismo de acesso aos casos que permite a manipulação dos dados. As demais formas de inspeção, que examinaremos logo adiante, pressupõem um exame passivo, geralmente realizado por ferramentas que fazem acesso direto à base de dados subjacente, o que normalmente não permite o acesso aos casos para agregação de trabalho.

Esta estratégia pressupõe que só devem ter acesso a um caso os agentes em cujas listas de afazeres constar uma atividade relacionada a este caso. A observação e a execução de atividades não previstas em uma especificação ficam, se não impedidas, prejudicadas.

2) As ferramentas de inspeção **por caso** são muitas vezes destacadas como sendo importantes, sem porém que haja uma discussão mais aprofundada de seu funcionamento, o mesmo ocorrendo em relação à inspeção **total,** que combina a de participante e caso em uma só, permitindo a inspeção de um conjunto de casos. Em Regatta, uma ferramenta externa acessa diretamente o banco de dados Sybase subjacente, em intervalos de 15 minutos, gerando referências exportadas para XMosaic, a partir do qual se pode ganhar acesso

aos colóquios [SMM+94] (veja Espaço compartilhado de participação) e supõe-se que, através destes, aos casos.

3) A inspeção de **segunda ordem** corresponde a uma análise estatística *post-hoc*, geralmente com objetivo de otimização. Em InConcert, diversas aplicações externas permitem a geração de relatórios e acompanhamento do andamento de tarefas e modificação de documentos, também através de acesso ao banco de dados [AS94]. Os relatórios "permitem a identificação de áreas onde melhorias de produtividade são possíveis" [Inc96b].

Um papel especial de alguns agentes é o de observador. O observador tem como tarefa básica se manter a par do andamento dos casos, não participando, porém, da realização das tarefas propriamente ditas [Ell95]. O observador intervém apenas em casos de exceção, ou se antecipa, para evitar que alguma exceção venha a acontecer. O observador tem um papel similar ao do piloto que se mantém atento ao painel de controle, mesmo quando o avião se encontra em automático, pronto para intervir sempre que necessário. Um observador pode reatribuir tarefas, modificar data limite, replanejar a divisão em etapas e qualquer outra ação corretiva que se faça necessária. É preciso que os mecanismos de inspeção possam por um lado suprir as necessidades de informação dos observadores e por outro permitir que eles atuem sobre os casos em que diagnosticarem problemas.

5.7. SUPORTE AO INDIVIDUALISMO

Vamos passar agora a discutir um novo aspecto, o do suporte ao individualismo. Este requisito implica na necessidade de se dar liberdade aos agentes para que implementem estratégias próprias diferenciadas para atingirem os objetivos de cada atividade. A necessidade de suporte ao individualismo vem da observação, aliás bastante evidente, de que grupos e indivíduos diferentes executam a mesma tarefa de formas variadas. Esta observação está relacionada ao fato de que não existe um único modo correto inquestionável de se atingir objetivos, mas diversos.

Da mesma forma que não existe um plano único, também não existe um conjunto de informações que seja o ideal. Principalmente em atividades menos estruturadas, cada agente terá o seu modo particular de trabalho, que se refletirá na necessidade de manipulação de conjuntos diferentes de informação a cada momento. A mesma atividade de análise de candidatos ao programa de pós-graduação, por exemplo, quando realizada por professores diferentes, pode exigir um conjunto diferente de informações, de acordo com a preferência de cada revisor: um deles pode desejar ver apenas as informações históricas de candidatos da mesma instituição, um segundo pode fazer esta pesquisa de acordo com os autores das cartas de recomendação e assim por diante.

1) Alguns sistemas (Regatta [SMM+94, SIM+94] e wOrlds [BK95], p.ex.) permitem que cada agente possa definir trechos de planos a serem executados em resposta a solicitações de seus serviços. O plano efetivamente utilizado durante a execução de um workflow resulta da composição dos planos individuais, i.é, a mesma atividade, se executada por dois agentes diferentes, obedecerão a sequências também diferentes.

2) Quanto ao individualismo relacionado aos dados, uma das raras referências é encontrada em [BN95]. Novamente, a ênfase no aspecto sincronizador faz com que este aspecto importante seja relegado a um segundo plano.

A discussão do individualismo envolve dois requisitos adicionais, o da privacidade e da customização de interfaces de usuário, que discutimos a seguir.

5.7.1. PRIVACIDADE

A garantia à privacidade das informações pessoais é um dos fatores de preocupação dos usuários de ferramentas de groupware (conforme [Orl92], [KHW93], [AS94], p.ex). Orlikowski apresenta em [Orl92] como fator que dissuade certos usuários da utilização de um sistema de groupware justamente o fato de que certas informações são consideradas por estes usuários como um patrimônio seu, já que resultam de investimento de tempo e esforço na sua coleta e organização, muitas vezes dispendido fora das horas normais de trabalho.

Para estes usuários, a visibilidade de suas informações privativas deve poder ser limitada, de forma que apenas eles próprios tenham acesso. Se um usuário fez uso de informações pessoais para tomada de decisão em certa atividade de um caso, apenas ele deverá poder enxergar estas informações, mesmo que esteja verificando um dado histórico, do passado de um caso, podendo reavaliar o motivo pelo qual tomou a decisão, enquanto que os demais agentes não vêem estas informações.

Swenson et al. mencionam em [SIM+94] que os planos, trechos de especificação sob responsabilidade de um agente, devem poder ser protegidos de forma similar, pois são vistos e com razão como vantagem competitiva. Saastamoinen e White relatam algo similar em relação a regras aplicadas pelos agentes: "regras individuais não são tornadas públicas; a sua existência pode não ser conhecida por nenhum outro agente, a não ser pelo seu criador, que provavelmente esconderá ou até negará a sua existência"[13] [SW95 p.304].

Quanto a garantia de privacidade dos dados, não são encontradas referências na literatura, salvo nos textos já mencionados dos autores ligados à etnografia, como Orlikowski [Orl92].

5.7.2. CUSTOMIZAÇÃO DA INTERFACE DE USUÁRIO

A interface de usuário da maioria dos sistemas é fixa, dando pouca ou nenhuma margem a customizações pelos usuários, que precisam se adaptar ao sistema, e não o contrário. O perigo desta abordagem é novamente o de que os projetistas suponham que determinada interface que lhes agrada irá necessariamente agradar aos usuários.

1) Alguns textos ([EW94b], [HKE92], [BN95], p.ex.) mencionam a vantagem de se permitir que os usuários possam customizar suas interfaces. Esta discussão se dá geralmente em termos de componentes fixos do sistema, como o gerenciador de tarefas, que apresenta normalmente uma lista de afazeres (*to-do list*), permitindo eventualmente que o usuário determine a ordem de apresentação dos itens, por exemplo (veja 5.9.1.-INSPEÇÃO).

2) Kyng expressa em [Kyn95] a posição de que a interface (e as funcionalidades de mais alto nível) de um sistema devem ser situadas em relação à comunidade de seus usuários, construído através de um processo de mútuo entendimento entre os técnicos e os usuários.

---

[13] *"Individual rule bases ... are not publicly available. Their existance may not even be known by anyone other than the actor using them. The actor may even hide or deny their existence"*

Um sistema deve oferecer, segundo esta visão, um conjunto aberto e extensível de mecanismos cuja operação possa ser realizada a partir de diversos modelos alternativos de interface, adaptados à realidade da comunidade que os utilizará e também passível de evolução.

Sistemas de workflow constituem necessariamente arcabouços, de preferência flexíveis, sobre os quais o verdadeiro sistema é construído, adaptado à realidade do momento da organização. Novamente o desafio consiste em se achar o equilíbrio ideal entre possibilidade x esforço de customização. Um sistema completamente amorfo exigirá um grande esforço de implantação, enquanto que um sistema rígido se torna menos útil.

5.8. FUNCIONALIDADE ADICIONAL

5.8.1. SUPORTE À EVOLUÇÃO

A necessidade de suporte à evolução é resultado da modificação constante apresentada pelas próprias organizações [EKR95, Saa94]. Para se manterem adaptadas à realidade externa à qual precisam reagir, as organizações devem a todo momento atualizar os seus processos. Quando estes processos estão automatizados, existe o risco de que as especificações existentes "engessem" o sistema, tornando-os menos efetivos ou inoperantes.

Um fator adicional a ser considerado é que a necessidade de mudanças globais são normalmente sentidas como contingências durante a execução de especificações existentes: mais e mais casos apresentarão problemas semelhantes, sinalizando que uma mudança geral ocorreu e que todas as especificações daqui para a frente devem passar a incorporar as novas características. Existe portanto um entrelaçamento entre 1) execução; 2) adaptações para fazer frente a contingências de casos e 3) adaptações gerais relativas a mudanças da realidade externa como um todo.

A estratégia comum neste caso é a de se modificar as especificações gerais a partir da qual os casos específicos são gerados. A modificação destas especificações podem ter diferentes impactos sobre os casos em execução nelas baseadas:

1) Nenhum impacto. Os casos em andamento se mantém seguindo de acordo com a versão existente no momento da criação da instância do caso. Este procedimento é adotado pelo sistema Regatta [SMM+94] e ActionWorkflow [MWF93], por exemplo.

O problema desta estratégia é que os agentes precisarão trabalhar sob diversos planos diferentes para alcançarem os mesmos objetivos, dependendo do momento em que o caso foi criado. Especialmente em processos de longa duração (meses ou anos), os planos locais acabarão tendo que sofrer replanejamento um a um, de forma manual, para igualá-los à especificação geral modificada.

2) Todos os casos passam a acompanhar a nova versão [EKR95]. O problema neste caso é que a nova versão pode não ser aplicável diretamente em todas as situações, por introduzir alguma incompatibilidade, ou simplesmente por não serem aplicáveis ao caso específico, como em um contrato, por exemplo, que deve seguir as regras acordadas inicialmente até o seu final).

3) Um esquema misto como o proposto por Bogia e Kaplan em [BK95] pode ser adotado. É proposto o uso de um mecanismo (*surrogates*) que permite que se gradue, caso a caso, trecho a trecho, qual a versão de especificação deve ser utilizada. É possível neste caso se determinar que o trecho inicial de um caso continuará a obedecer uma versão determinada de especificação, enquanto que o trecho final acompanha a última versão. O mecanismo é controlado por regras, podendo-se alterar as versões utilizadas dinamicamente, em tempo de execução.

Concluímos que o suporte à evolução deve ser homogêneo ao utilizado no tratamento de qualquer contingência local, que por sua vez deve ser o mais semelhante à realização do trabalho normal, como discutido em 5.7.3.-PLANEJAMENTO COLABORATIVO. Das três estratégias de adaptação a novas especificações, a proposta de Bogia e Kaplan é evidentemente a melhor, por permitir que se escolha com uma granularidade mais refinada qual a versão de especificação a se utilizar.

5.8.2. PLANOS ALTERNATIVOS

Conforme ocorrem as contingências, novas variantes de especificações são criadas, adaptadas às suas especificidades. Em um dado momento, a um determinado objetivo corresponderão normalmente um grupo de especificações relacionadas, e não apenas uma.

A existência destas famílias de especificações relacionadas é reconhecida em diversos textos ([SW95], [AS94], [SMM+94], p.ex.). Existe uma clara necessidade de se organizar estas especificações, agrupando-as para reuso, destacando-se as diferenças entre elas, de forma a permitir a seleção da variante mais apropriada a cada situação.

Em Tosca, o mecanismo básico de gerenciamento de recursos organizacionais é utilizado para unir *templates* relacionados através de *hiperlinks* [Pri93 p.149]. O sistema Chautaqua agrupa as variantes ao apresentá-las em um *browser* de planos [Ell95]. Uma proposta mais ambiciosa é apresentada por Malone et al. em [MCL+92]. Esta pesquisa aborda o problema das especificações a partir de um outro ângulo, o da construção de um manual universal de *templates* de processo, que pode ser utilizado tanto para a análise de alternativas de implementação de um mesmo processo básico quanto da inovação de processos através da combinação original de perspectivas de administração. Relevante para a nossa discussão é a construção de novos modelos por herança e os mecanismos de explicitação de similaridades e diferenças adotados.

5.8.3. SCRITPS E AÇÕES AUXILIARES

Diversos sistemas oferecem a possibilidade de disparo de *scripts* associados à ativação ou desativação de atividades. Pouco se fala das utilizações pretendidas em cada sistema, mas pode-se supor que funcionam como mecanismo aberto que permite a especificação de ações preparatórias e de finalização não previstas diretamente no sistema. Através de *scripts* disparados automaticamente, alguma funcionalidade pode ser adicionada. Novamente, diversos mecanismos diferentes são oferecidos:

1) O sistema Regatta, por exemplo, permite a utilização de *scripts* associados a opções dos estágios. Assim que uma opção é selecionada pelo usuário, se definido, um *script* é ativado, o que permite a introdução de "efeitos colaterais adicionais à ativação de estágios" [SMM+94 p.23].

2) No sistema wOrld, alguns eventos podem iniciar a execução de *applets*, trechos de código Smalltalk. Os eventos estão associados à criação e destruição de *locales*, adição de novo participante em um *locale* e a mudanças de estado sinalizadas por opções dos usuários [TKF95 p.60-61].

3) Em InConcert [MS93] e ActionWorkflow [MWF93] regras são utilizadas para disparo de ações, baseadas em temporizadores ou outros eventos. O uso mencionado inclui o disparo automático de workflows e envio de notificações (veja o tópico). InConcert provê um mecanismo extensível que permite que novos eventos possam ser definidos e tratados [MS93]. Tanto em InConcert quanto em ActionWorkflow, estas regras constituem um mecanismo paralelo, independente do sincronismo normal, baseado em *t-actions* (veja 4.5.-DISPARO DE ATIVIDADES).

4) Em $Trigs_{flow}$, uma camada ativa do banco de dados é responsável pelo tratamento de qualquer evento, seja relacionado ao sincronismo, seja ao disparo de ações auxiliares [KLR+95].

Por oferecer um mecanismo genérico capaz de funcionar tanto como *script* como quanto plano de disparo de atividades, regras nos parecem boas candidatas a formarem o mecanismo básico reativo de um sistema. Um cuidado necessário neste caso é o de conversão da representação expressa nas regras para alguma notação gráfica mais simples, manipulável pelos usuários.

5.8.4. HISTÓRIA DE ATUALIZAÇÕES

Uma fonte importante de informação é a relacionada ao histórico das atualizações. Algumas vezes, especialmente em situações de tratamento de contingências, o exame dos valores atuais dos objetos pode não ser suficiente. A seqüência de modificações e o registro de quem as executou formam uma história que, se disponível, pode auxiliar no futuro no diagnóstico de algum erro realizado ao longo do processamento, só percebido mais tarde.

Além do registro automático, é preciso que se possa examinar esta história e até restaurar os estados pretéritos de objetos, descartando seletivamente modificações feitas a partir de determinado momento, de forma automática ou manual. Este mecanismo corresponde de alguma forma ao *undo/redo* disponível em algumas ferramentas.

1) No sistema Regatta, um mecanismo de *undo* primitivo permite apenas o reinício de execução de um *node*, com o abandono de todas as modificações realizadas [SMM+94].

2) Em InConcert, são mantidos históricos de versão (*version histories*). Neste sistema, é permitida consulta e o *roll-back* manual, sob controle do usuário [MS93].

3) [SIM+94] e [AS94] mencionam a necessidade de se prover suporte a versões de processo, o que é bastante razoável se considerarmos que a evolução natural das organizações fará com que as especificações sofram alterações constantes. A evolução temporal do esquema em geral, incluindo as especificações de processo, é fundamental para que se possa examinar dados históricos no contexto em que ocorreram, que pode ser consideravelmente diferente do atual.

5.8.5. SUPORTE A APLICAÇÕES EXTERNAS

Como não se pode conhecer a *priori* quais são os tipos de objetos e ferramentas utilizados em um ambiente, é preciso que exista um mecanismo que permita a inclusão de novas ferramentas e seu uso integrado na realização de tarefas do sistema.

1) Kreifelts et al. mencionam em [KHK+91] o desejo dos usuários em poderem usar as ferramentas (editores e planilhas, p.ex.) aos quais estão acostumados para a realização das tarefas associadas ao workflow. Esta noção é compartilhada por alguns outros autores ([AS94], [MS93], [HKE92], p.ex.).

2) No sistema WooRKS [ALP+94], cada usuário pode determinar suas preferências em relação a ferramentas externas. Desta forma, o mesmo documento texto pode ser manipulado por um agente através do Emacs, enquanto que um segundo utiliza vi, por exemplo.

5.9. COMPARATIVO

A tabela da figura 32 apresenta a distribuição de implementações (I) e de menções (x). As marcações com "I?" indicam que uma implementação parcial ou questionável é oferecida.

Os artigos estão agrupados por sistema ou por autor principal, no caso de teóricos.

	SIM+94 SMM+94 SI95	AS94 Inc96a Inc96b MS93	HKE92 KHW93 KHK+91 Pri93	FTK95 TKP95 BK95	MWF93 Med92 Flo96	KLR+95 Bus94	CCP+95a CCP+95b	EW94a EW94b El95 EKR95	BR94	Rob93 RB91 Suc87 Suc93 Suc95	BW95a BW95b WB96	Outros Textos	Imp	Menção
Integração do planejamento e execução	I?	x	I	I				x		x	x	Hir86,Saa94,Sac95	3	7
Ações ad-hoc e especificações								x		x	x	Gal77		4
Maior poder de especificação							x				x			2
Eventos assíncronos	I?	x			x	I	I?				x		3	3
Atividades batch											x			1
Ativações paralelas	I?	x					I						2	1
Ativações baseadas em dado	I?		I	I		I	I						5	
Seleção de executores mais poderosa						x	x				x			3
Ambiente de execução	I?	I?	I	I				x			x	Kar90,Su93	4	4
Inspeção	I	I	I	I	I			x					5	1
Awareness			x	I				x	x	x			1	4
Comunicação			I	I				x	x	x			2	3
Planejamento colaborativo	I?	x	x	I				I			x		3	3
Suporte ao individualismo	I			I				x			x		2	2
Privacidade	I	I	x									Ori92	2	2
Customização da Interface			x					x	x			Kyn95		4
Funcionalidade adicional	--	--	--	--	--	--	--	--	--	--	--			
Suporte à evolução	I		I	I	I	I	x	x			x		5	3
Planos alternativos	x	x	I					I			x	SW95, MCL+93	2	5
Scripts e ações auxiliares	I	I	I	I	I	I							6	
História de atualizações		I							x		x		1	2
Suporte a aplicações externas		I	I		I							ALP+94	3	1
Implementações	11	6	9	10	4	4	3	2					49	--
Menções	1	5	4		1	1	3	9	4	4	12	11	--	55

Figura 32 - Referência cruzada entre requisitos desejáveis e artigos.

5.10. CONCLUSÕES

Analisamos requisitos específicos e o modo como são encarados em outros sistemas, a partir da perspectiva de equilíbrio entre elementos fundamentais, correspondentes ao aspecto guardião, comunicador e sincronizador, além da oferta de ações *ad-hoc* e especificações sem impedância.

Os requisitos apresentados destacaram os pontos em que a maioria dos sistemas são omissos ou oferecem funcionalidade aquém do desejável, principalmente relacionadas a:

- Integração de planejamento e execução;
- Ações e especificações mais poderosas;
- Ambientes de execução;
- Suporte ao individualismo;
- Funcionalidades adicionais.

A partir dos requisitos levantados, proporemos no próximo capítulo um modelo próprio, que procura atendê-los de forma mais abrangente do que a maioria dos sistemas atuais.

6. UM MODELO CONCEITUAL DE WORKFLOW

6.1. INTRODUÇÃO

Baseados na discussão dos capítulos 3, 4 e 5, vamos agora definir o nosso próprio modelo, cuja ênfase é a abrangência, em detrimento do detalhe: procuramos determinar de forma ampla uma base conceitual de um gerenciador de workflow que, esperamos, possa servir de fundamento para pesquisas futuras, especialmente as relacionadas ao gerenciamento de dados, que corresponde justamente ao aspecto guardião, que não é normalmente levado em consideração em sistemas de workflow atuais.

Os problemas são atacados sob quatro perspectivas:

- Aumento da expressividade - são propostas ações e elementos de sincronismo mais poderosos, capazes de expressar situações para as quais sistemas existentes não oferecem suporte;
- Ambiente de execução - a execução do trabalho, tanto de manipulação de dados de aplicação quanto dos referentes ao próprio sistema, as especificações, são contemplados através de uma abordagem uniforme, orientada a dados;
- Alocação de executores - propomos uma estratégia abrangente para o problema da seleção dos agentes mais apropriados para a execução de cada atividade, crucial em sistemas de workflow;
- Comunicação - a comunicação é encarada no contexto mais abrangente da difusão da *awareness*, que se preocupa em manter cada agente a par das ações dos demais, proporcionando a sinergia, essencial em sistemas colaborativos.

Podemos perceber que todos os elementos básicos, referentes aos aspectos guardião, sincronizador e comunicador são contemplados em nossa proposta de modelo. Mais ainda, como teremos ocasião de perceber, o modelo reflete a preocupação com a execução de ações *ad-hoc*, com a adaptabilidade, capacidade de evolução e suporte ao individualismo, que estabelecemos como requisitos no capítulo 5.

Apresentaremos inicialmente uma discussão sobre as ações e sincronismo (6.2), seguida pelos conceitos relacionados ao ambiente de execução (6.3). Em 6.4.-SELEÇÃO DE EXECUTORES, apresentamos a estratégia de escolha de agentes e em 6.5.-DIFUSÃO DE AWARENESS são discutidos os aspectos de comunicação. O capítulo encerra com conclusões.

6.2. AÇÕES E SINCRONISMO

Trataremos inicialmente do aumento de expressividade do modelo, através da oferta de ações e elementos de sincronismos mais poderosos do que os comumente encontrados. Este aumento de expressividade se dará pelo tratamento de eventos assíncronos, atividades batch e atividades replicadas. O conjunto de ações propostas procura evitar a impedância entre as ações constantes de especificações e as realizadas de forma *ad-hoc*, podendo ser utilizadas indistintamente nos dois modos.

Uma tarefa básica de um sistema de gerenciamento de workflows é a de permitir que as etapas necessárias à obtenção de determinado objetivo de negócio possam ser ativadas de forma automática, a partir de uma especificação que estabelece uma ordenação parcial entre estas etapas, havendo a distribuição destas tarefas para diversos agentes.

O **plano de processamento** é o componente que armazenará as especificações dos processos, estabelecendo o momento adequado de disparo das ações relacionadas ao processo. Dos diversos tipos de ações que podem ser disparadas (que examinaremos em 6.2.3.-AÇÕES), uma categoria nos interessa em especial, a das ações de disparo de atividade. As atividades são as unidades de execução de trabalho do sistema, envolvendo um ou mais agentes.

O objetivo geral de um processo, o de conduzir a seleção de candidatos à pós-graduação, por exemplo, normalmente é subdividido em etapas mais simples, em atividades, individuais ou coletivas, que tem um objetivo próprio. Quando uma atividade é muito complexa, é conveniente que se possa subdividí-la, por sua vez, em sub-atividades mais simples, de forma semelhante ao que acontece com um processo.

Ao encontrar uma sub-atividade que possui plano próprio, o sistema se encarrega de compor o plano geral (que chamaremos de super-plano) com o sub-plano, substituindo a atividade mais abstrata pelos elementos expressos no sub-plano (fig. 33). Examinaremos este mecanismo em mais detalhes em 6.3.2.8.-LÓGICA DE ATIVAÇÃO DE CONTEXTOS.

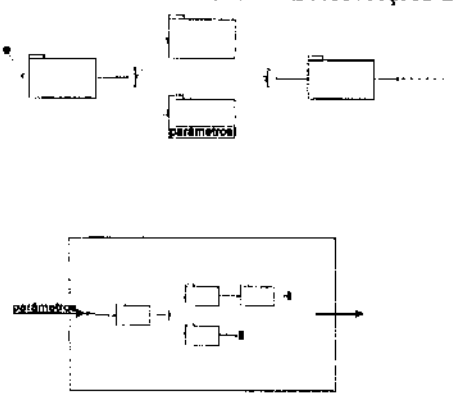


Figura 33 - Composição de planos.

Esta decomposição pode se repetir um número ilimitado de vezes, ou seja, uma sub-atividade também pode ter uma subdivisão. Pode-se, portanto, construir hierarquias de abstração com quantos níveis forem necessários. Obviamente, um número excessivo de subdivisões pode não ser desejável por motivos metodológicos, da mesma forma que não é desejável a subdivisão excessiva em uma hierarquia de módulos de um programa, ou em um sistema convencional.

O plano de processamento estabelece um roteiro que será seguido pelas instâncias reais do processo, que chamamos de **casos**. Na seleção de candidatos à pós-graduação, por exemplo, existirá um caso para cada um dos candidatos que se inscreverem. Cada um destes casos obedece ao ciclo de vida especificado no plano de processamento, salvo eventuais exceções, que imporão alterações de curso individuais.

A cada momento, existirão potencialmente diversos casos de um mesmo processo, transcorrendo em paralelo. Cada caso é independente dos demais, e cada um deles pode se encontrar em uma fase diferente. O mecanismo de sincronismo se encarrega de monitorar a ocorrência de eventos de cada caso, disparando ações e notificando agentes, conforme o especificado no plano de processamento e o estado atual dos dados do caso (fig. 34).

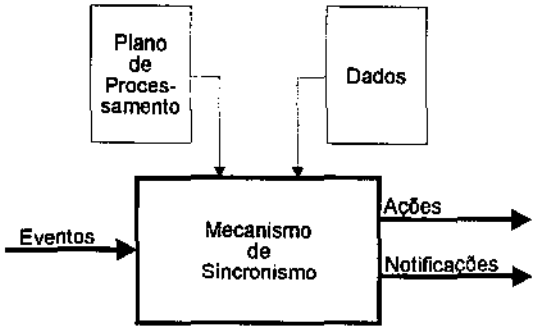


Figura 34 - Mecanismo genérico de sincronismo.

Note que ao admitirmos a influência dos dados no mecanismo de sincronismo, já estamos tomando uma posição diferente da maioria dos sistemas existentes, que costumam ser comandados apenas pela ocorrência de eventos gerados explicitamente pelos agentes, através da ativação de itens de menu ou outro elemento qualquer de interface (veja 4.5.- DISPARO DE ATIVIDADES).

O plano de processamento é constituído por uma malha com três tipos de elementos básicos:

- Eventos - sinalizam acontecimentos que exigem reação do sistema, como o início de um novo caso, o término de atividades em execução e assim por diante;
- Elementos de sincronismo - estabelecem as dependências entre ações;
- Ações - correspondem ao processamento em si, normalmente associado ao disparo de atividades.

Vamos examinar cada um destes elementos em maiores detalhes a seguir.

6.2.1. EVENTOS E *THREADS*

Basearemos o mecanismo de disparo do nosso modelo em eventos. Podemos distinguir dois tipos básicos de eventos: os que dão início a uma seqüência nova de processamento e os que dão continuidade a seqüências já existentes.

1) Eventos iniciadores - o mais básico destes eventos é o gerado automaticamente sempre que um novo caso é criado. Este evento tem por função colocar em andamento o processamento do caso. Na maioria dos sistemas existentes, este é o único evento iniciador existente.

Representaremos o evento inicial pela figura 35, e diremos que ele dá início ao ***thread* default**, que é ativado de forma automática sempre que for criada uma nova instância de caso. Os parâmetros, se especificados, permitem que sejam fornecidas informações iniciais de processamento.

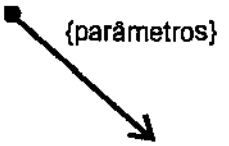


Figura 35 - Evento inicial.

Discutimos em 5.5.1.-EVENTOS ASSÍNCRONOS, a existência de outros tipos de eventos que ocorrem opcionalmente, de forma assíncrona. Estes eventos sinalizam situações como solicitações externas de cancelamento (vindas de um cliente, p.ex.), alterações imprevistas de informação, expirações de prazos limite e assim por diante. Não se pode prever se estes eventos ocorrerão ou não, e nem o momento em que virão a acontecer. Quando um destes eventos ocorre, eles devem dar início a uma nova seqüência de tratamento.

Eventos assíncronos são geralmente tratados como exceções, fora do escopo das especificações e, muitas vezes, fora do escopo do sistema como um todo. Em nosso entender, a impossibilidade de se poder expressar este tipo de evento é um impecilho grave, por obrigar os usuários a tratá-los manualmente, mesmo em processos em que a sua incidência é alta. Isto é evidentemente menos ótimo do que o tratamento conduzido sob controle do próprio sistema.

Permitiremos que sejam especificados *threads* adicionais, além do default, dedicados ao tratamento de cada um destes eventos opcionais. O plano de processamento é, portanto, um grafo que possui potencialmente diversos componentes conexos, referentes aos *threads*, cada qual estabelecendo a reação a um eventos iniciador. A representação de eventos deste tipo pode ser vista na figura 36.

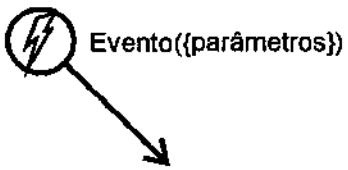


Figura 36 - Eventos assíncronos.

Nossa solução permite que qualquer ação ou seqüência de ações possa ser utilizada em resposta a um evento opcional. A reação a um evento deste tipo pode, portanto, ser tão complexa quanto o necessário, constituindo, se desejado, workflows paralelos de tratamento (fig. 37).

Os eventos opcionais, caso ocorram, iniciam **em paralelo** um novo *thread*, que processa este evento. As ações de manipulação de ações correntes e de *threads*, como veremos, podem ser utilizadas para suspender ou cancelar qualquer outra ação em execução que interfira com o processamento desejado. No caso de um pedido de cancelamento, por exemplo, pode-se determinar a suspensão das ações correspondentes ao *thread* default, até que se possa avaliar a situação.

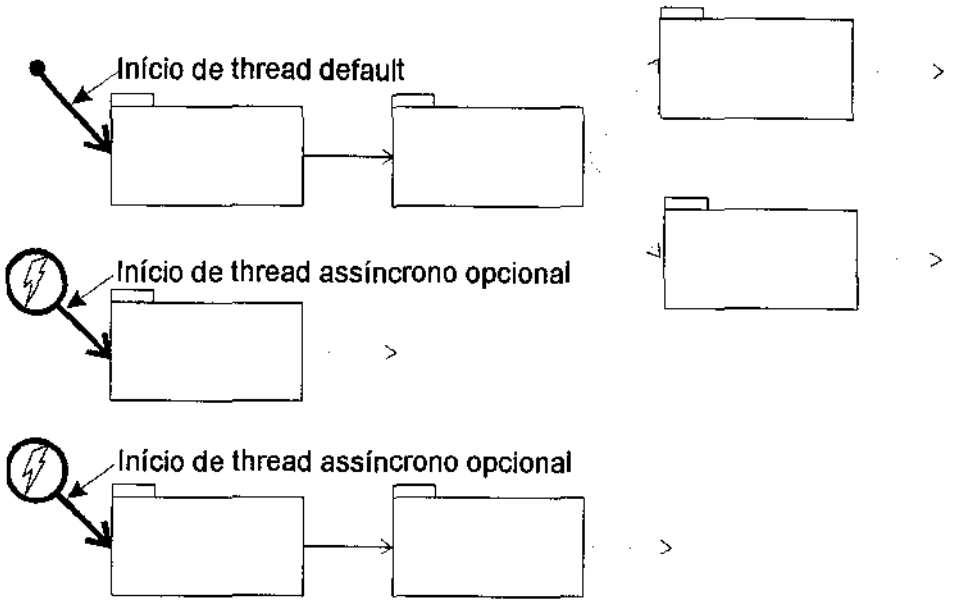


Figura 37 - *Threads* de processamento.

2) A segunda categoria de eventos se refere àqueles que dão continuidade a *threads* já em andamento. As *t-actions* [EW94a] são os principais eventos deste tipo, e sinalizam que uma determinada ação disparada anteriormente pelo sistema terminou e que o caso pode fluir adiante. Esta terminação ocasionará potencialmente o disparo de outras ações dependentes do término desta ação anterior. São representados por arestas que ligam os elementos do grafo (fig. 38).

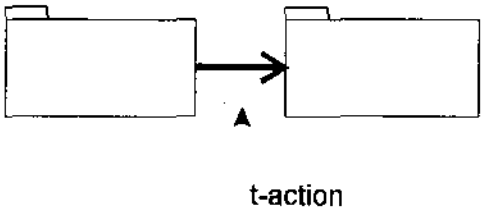


Figura 38 - Representação de *t-actions*.

Por fim, temos os eventos esperados, que normalmente correspondem a ocorrências que sinalizam acontecimentos do mundo real, como a chegada de uma mercadoria, ou a liberação de uma sala de reunião, por exemplo, esperados durante o processamento de um caso. A não ocorrência de um evento esperado constitui uma contingência. Utilizaremos para estes eventos a mesma representação da figura 36, com a diferença importante de que estes eventos aparecem no corpo do *thread*, e não no seu início (fig. 39).

Uma classe especial é formada pelos eventos de temporização, associados ao transcurso do tempo. Caso uma ação só deva ser disparada após um momento específico, por exemplo, um evento de tempo pode ser utilizado. As representações que utilizaremos para temporizadores pode ser vista na figura 40.

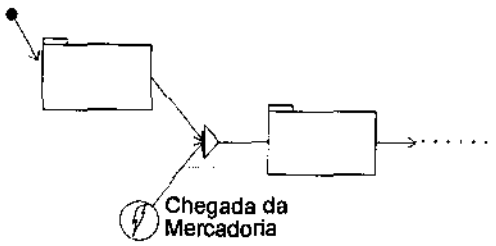


Figura 39 - Eventos esperados.

Eventos de temporização também podem dar início a *threads* independentes, por exemplo, quando se deseja iniciar novo processamento em uma data determinada, ou periodicamente.

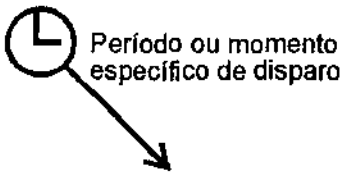


Figura 40 - Representação de eventos e temporizadores.

Todos estes eventos apresentados, especialmente as *t-actions*, são utilizados em conjunto com elementos de sincronismo, que estabelecem as condições de disparo de cada ação. Examinaremos estes elementos a seguir.

6.2.2. ELEMENTOS DE SINCRONISMO

Estes elementos permitem que sejam especificadas as dependências entre as ações de cada *thread*, ou seja, as restrições de que certas ações só podem ser disparadas após o término de outras de que dependem, como a distribuição para avaliação no processo de seleção de candidatos à pós-graduação, por exemplo, que só pode ter início depois que a inscrição foi completada.

Os elementos que passamos a apresentar buscam atender os requisitos determinados em 5.5.-MAIOR PODER DE ESPECIFICAÇÃO, oferecendo recursos mais poderosos de sincronismo, principalmente os relativos a atividades batch, ativações paralelas e ativações condicionais baseadas em dados, que vimos serem úteis mesmo na especificação de processos simples como os apresentados em 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO.

As etapas de processamento são executadas em seqüência se houver dependência entre elas, podendo ser executadas em paralelo, caso contrário. Atividades paralelas e alternativas introduzem a necessidade de se poder explicitar a divisão e junção.

A idéia básica é a de que as etapas só devem ser disparadas quando algumas pré-condições tiverem sido satisfeitas. Estas pré-condições se baseiam principalmente na terminação de uma ou mais etapas de processamento anteriores. Os elementos de sincronismo podem ser vistos como portas lógicas que habilitam uma ou mais conexões de saída dependendo de condições de entrada. Esta habilitação implica no disparo de novas ações.

Passaremos agora à especificação dos mecanismos de nosso modelo, adaptados a partir da proposta de Casati et al. [CCP+95a]. Os elementos que utilizaremos são os seguintes:

Forks - estes elementos de divisão podem ser vistos como portas lógicas que multiplicam um sinal recebido, dependendo do seu tipo:

- Total - habilita todas as conexões que partem do *fork*, ou seja, todas as ações subseqüentes serão disparadas em paralelo (fig. 41-a).
- Condicional - identifica as condições que ocasionam o disparo relativo a uma conexão. São habilitadas as conexões que corresponderem a condições verdadeiras (fig. 41-b).
- Replicação - esta forma especial de divisor ocasiona a replicação dos elementos que o sucedem, de acordo com um número *k* especificado (corresponde à *multitask* de [CCP+95a]). A representação deste elemento é apresentada na figura 41-c.

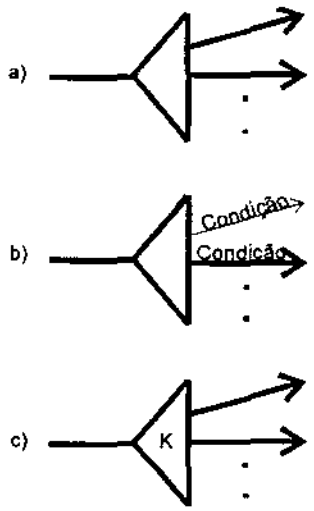


Figura 41 - a) *fork* total b) *fork* condicional c) replicação.

Joins - propiciam o complemento dos *forks*, recebendo diversos sinais de entrada e gerando apenas um de saída.

- Total - todas as entradas devem estar habilitadas para que a conexão de saída seja habilitada (fig. 42-a);
- Parcial - permite a especificação de um *quorum k*, habilitando a saída quando este número de sinais de entrada estiver habilitado. Os demais sinais, recebidos após a ativação inicial, são ignorados (fig. 42-b). Caso se especifique *k* igual a 1, obtemos a semântica de um *or-join* [WMC95], i.é, a saída é habilitada assim que o primeiro sinal de entrada for habilitado, e os demais são ignorados.
- Iterativo - cada vez que *k* sinais de entrada estiverem habilitados, habilita um sinal de saída. Equivale a um *join* parcial em que a contagem de sinais habilitados é retornada a zero a cada vez que o sinal de saída for habilitado, permitindo novos disparos cíclicos (fig. 42-c).

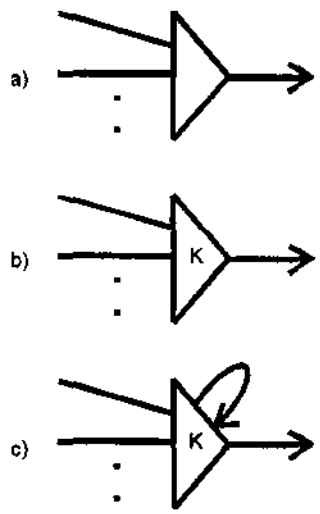


Figura 42 - a) *join* total b) *join* parcial (*quorum*) c) *join* iterativo.

Batch - A principal extensão de nosso modelo em relação aos elementos de sincronismo usuais é relativa à sincronização de atividades batch, conforme [BW95a, BW95b]. Este tipo de sincronismo, apesar da sua utilidade prática, como já tivemos ocasião de mencionar, não é oferecido em qualquer sistema do qual tenhamos conhecimento, e nem é mencionada em outros textos que não os de nossa própria pesquisa.

Em atividades batch, o sincronismo se dá em uma dimensão diferente, a dimensão dos casos. Um exemplo deve esclarecer a natureza deste sincronismo: para se poder realizar a atividade de classificação final no processo de seleção de candidatos à pós-graduação, por exemplo, é preciso que **todos** os candidatos já tenham sido avaliados por um número mínimo de revisores. Enquanto que o sincronismo expresso pelo *fork* e *join* se referem às etapas de **um** caso, o batch sincroniza a execução de etapas de **diversos** casos do mesmo tipo (fig. 43).

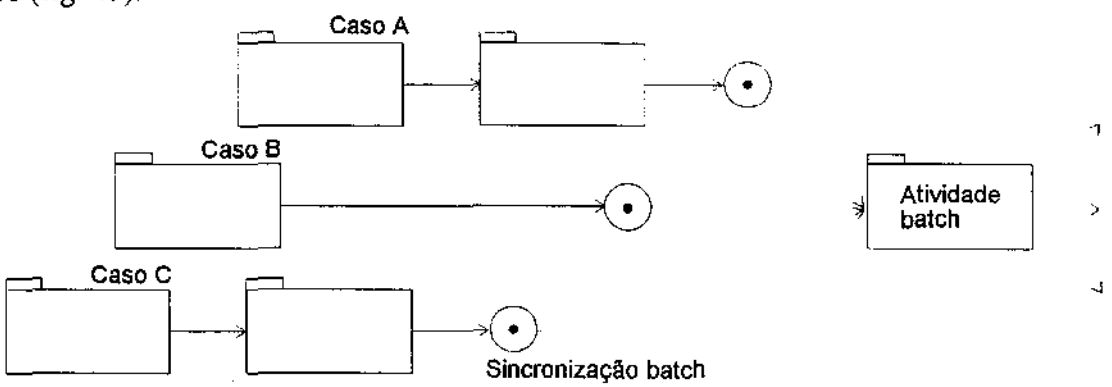


Figura 43 - Sincronização batch.

A atividade que segue uma sincronização batch utilizará como dados o agrupamento de todos os casos individuais. Cada caso mantém a sua identidade, e pode-se declarar a atividade correspondente ao batch como encerrada (gerando a *t-action*) de forma independente para cada um deles. Em outras palavras, os casos não precisam encerrar esta atividade no mesmo momento, e nem seguir o mesmo destino. A partir das manipulações realizadas

pelos executores da atividade batch, por exemplo, alguns casos podem ser aprovados e outros rejeitados, que é justamente o que acontece na seleção de candidatos à pós-graduação.

Identificamos dois tipos de batch:

- Total - indica que todos os casos deste tipo são sincronizados neste ponto (fig. 44-a).
- Iterativo - aguarda até que *k* casos cheguem a este ponto antes de habilitar a saída. Este tipo de sincronização permite que os casos sejam agrupados em lotes com número fixo de elementos (fig. 44-b).

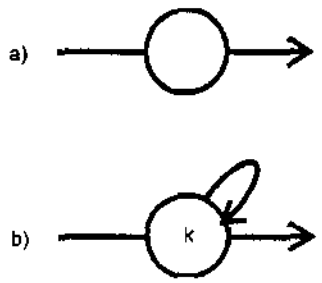


Figura 44 - a) batch total b) batch iterativo.

Vamos permitir que cada elemento de sincronismo especifique seus próprios tratamentos de eventos assíncronos opcionais, de forma semelhante ao proposto por Casati et al. em [CCP+95a, CCP95b]. Isto permite, por exemplo, que sejam especificados tratamentos para expiração de tempo máximo de execução (*time-out*) para cada um dos elementos. Se os sinais de que um *join*, *fork* ou *batch*, dependem para disparar não forem recebidos até determinada data, por exemplo, pode-se gerar uma notificação, ou cancelar o caso, ou forçar o prosseguimento e assim por diante (fig. 45). A semântica necessária para isto será examinada em 6.2.3.-AÇÕES.

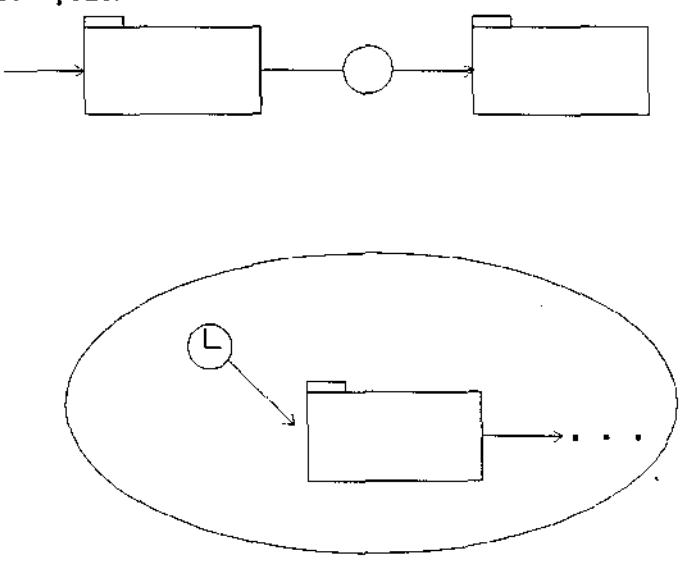


Figura 45 - Evento assíncrono opcional associado a elemento de sincronismo.

Mencionamos a existência de ações que permitem a emissão de notificação, cancelamento de atividade, de caso e assim por diante. Vamos examinar em seguida quais são estas ações.

6.2.3. AÇÕES

Vimos qual o papel dos eventos e dos elementos de sincronismo no plano de processamento. Vamos examinar agora quais são as ações que podem ser disparadas durante o processamento de um caso.

Tomadas em conjunto, estas ações representam a funcionalidade básica oferecida pelo modelo. Obedecendo à determinação de que não deve haver impedância entre a linguagem de especificação e as ações *ad-hoc*, estabelecida em 5.4.-AÇÕES *AD-HOC* E ESPECIFICAÇÕES, deve ser possível ativar qualquer destas ações tanto a partir de um menu, à disposição de agentes autorizados, quanto incluí-las em uma especificação. Neste último caso, as ações correspondem a uma representação gráfica. As informações necessárias à execução serão fornecidas no momento da ativação da ação: se realizada de forma *ad-hoc*, através de janela de diálogo; se sob controle do plano, como propriedades associadas às representações gráficas utilizadas na linguagem de especificação.

Podemos distinguir algumas categorias de ações:

- Ações básicas - correspondem a atividades e notificações, que constituem a essência do trabalho de processamento propriamente dito;
- Manipulação de atividades correntes - estas ações permitem que atividades em execução sejam suspensas, canceladas, reativadas e assim por diante;
- Manipulação de *threads* - todas as ações de um determinado *thread* podem ser controladas através destas ações;
- Manipulação de caso - se desejarmos manipular todas as ações do caso como um todo, utilizamos estas ações;
- Ações sobre dados - permitem que sejam incluídas nas especificações as ações sobre os dados, como a de criação de instâncias, modificação de valores de atributos, e assim por diante;

Vamos examinar cada uma das ações disponíveis, utilizando uma notação que especifica as informações necessárias e seus respectivos tipos como se fossem parâmetros em um cabeçalho de procedimento. Informações opcionais são exibidas em *itálico*.

6.2.3.1. Ações básicas

Estas são as ações fundamentais do modelo, que comandam a execução de trabalho dos agentes ou do próprio sistema.

- **Notificação(mensagem:tipomens, {descrição de agentes:string})**

Envia uma mensagem para os agentes especificados. Este mecanismo serve para solicitar a atenção dos agentes sobre alguma situação, geralmente anormal, sobre a qual eles devem tomar conhecimento, como por exemplo, a expiração da data limite de execução de uma determinada atividade anteriormente iniciada, ou qualquer informação *ad-hoc* gerada pelos agentes.

Toda notificação é automaticamente registrada pelo sistema, para referência futura, como teremos ocasião de examinar em 6.3.2.-CONTEXTOS DE EXECUÇÃO. A notação gráfica de notificações pode ser vista na figura 46.

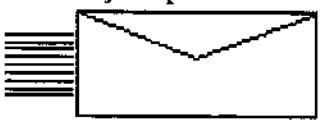


Figura 46 - Representação de notificações.

As descrições de agentes são baseadas no modelo organizacional, por exemplo, "João", "Funcionário", "Secretário do departamento de marketing". O sistema se encarrega de processar estas descrições (veja como em 6.4.-SELEÇÃO DE EXECUTORES), enviando a mensagem aos agentes correspondentes a elas.

- **Disparo de atividade(rótulo:string, atividade:string, *{parâmetros:identobj}*, *{descrição agentes:string}*, *n:número*, *estratégia:string*)**

O início de execução de uma atividade é provavelmente a ação mais comum realizada em resposta a eventos. A figura 47 mostra a representação gráfica que utilizaremos para identificar disparos de atividades.

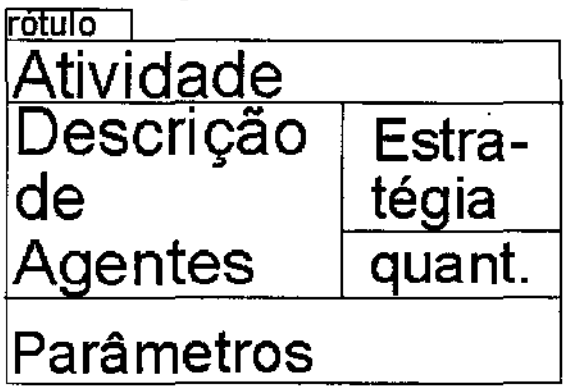


Figura 47 - Representação de atividades.

Vamos examinar cada uma das informações associadas ao disparo de uma atividade:

1) **Rótulo** - é utilizado para referências futuras à atividade, de forma a permitir que esta possa ser suspensa, ou cancelada, por exemplo, como veremos em 6.2.3.2.-MANIPULAÇÃO DE ATIVIDADES CORRENTES.

2) **Atividade** - o nome da atividade identifica o tipo de serviço a ser realizado. Este nome é utilizado para identificar o artefato específico que dará suporte à realização das tarefas relativas a esta atividade, como veremos em 6.3.2.-CONTEXTOS DE EXECUÇÃO.

3) **Identificação dos executores** - o modelo propicia uma alocação de executores mais abrangente do que a da maioria dos sistemas, que pressupõem que a execução de cada atividade é realizada por um único agente, escolhido através de uma estratégia fixa do sistema.

Consideramos que cada atividade pode ter um conjunto de executores que realizam as tarefas relativas a ela de forma colaborativa, síncrona ou assincronamente. Quando uma atividade deve ser iniciada, todos os agentes responsáveis pela sua execução são notificados

do fato. Como em nosso modelo os artefatos de suporte ao trabalho podem permitir trabalho síncrono ou assíncrono por mais de um agente, como examinaremos em 6.5.-DIFUSÃO DE AWARENESS, diversos agentes podem estar associados a uma única atividade sem maiores problemas, realizando-a de forma cooperativa, sob mediação dos artefatos do contexto. Com isto, pode-se especificar atividades de tomada cooperativa de decisão, por exemplo, o que não é normalmente possível em qualquer outro modelo.

Os executores são determinados pelos parâmetros "{descrição de agentes}, estratégia, n", onde:

- **Descrição de agentes** é um conjunto de descrições de agentes baseadas no modelo organizacional, similar à utilizada para a determinação dos agentes na emissão de notificação, por exemplo, "João", "Funcionário", "Secretário do departamento de marketing". Os agentes correspondentes a estas descrições são os candidatos em potencial a assumirem a execução da atividade. A definição exata de quais deles realmente assumirão a execução é decorrente dos itens **N** e **Estratégia** descritos abaixo;
- **N** estabelece quantos executores serão atribuídos simultaneamente à atividade. Podemos especificar, por exemplo, que os três primeiros candidatos selecionados de acordo com a estratégia determinada se encarregarão da execução efetiva da atividade, ou então que todos os agentes disponíveis serão utilizados.
- **Estratégia** descreve o critério de seleção, caso as descrições se apliquem a mais de agentes do que o necessário, ou seja, se existirem mais candidatos do que o especificado em **N**. Este critério corresponde a uma das estratégias oferecidas pelo sistema, como balanceamento de carga de trabalho, por turnos (*round-robin*) e assim por diante (veja 6.4.-SELEÇÃO DE EXECUTORES).

5) **Parâmetros** - Pode-se especificar para cada atividade os parâmetros que ela utiliza. Estes parâmetros são as variáveis de controle utilizadas na execução da atividade, ou geradas por ela. O objetivo é o de se deixar explícitos os locais em que estas variáveis são manipuladas.

Acabamos de examinar as ações básicas do modelo. Vamos examinar a seguir as ações de manipulação de atividades correntes, que permitem que se controle atividades em execução, de forma *ad-hoc* ou via especificação.

6.2.3.2. Manipulação de atividades correntes

Uma outra categoria de ações normalmente só executáveis de forma *ad-hoc* na maioria dos sistemas, é a que permite que se possa controlar a execução de atividades em execução em um determinado momento. Esta manipulação opera transições entre estados do ciclo de vida das atividades (fig. 48).

Uma atividade, assim que disparada, passa ao estado de **pronta**, aguardando que seus executores dêem início ao trabalho. Assim que isto acontece, a atividade passa a estar **ativa**. A atividade pode ser **encerrada, cancelada** ou **suspensa**. O encerramento corresponde à geração de uma *t-action*, sendo o estado terminal mais comum e indica sucesso na obtenção do objetivo da atividade. Atividades suspensas não admitem que seus executores continuem executando suas tarefas. Este estado é utilizado para evitar que o trabalho prossiga quando ainda não se sabe, em decorrência de alguma contingência, se ele será

útil ou não. O cancelamento é um estado terminal que indica que o objetivo da atividade não foi atingido, tendo havido um abandono.

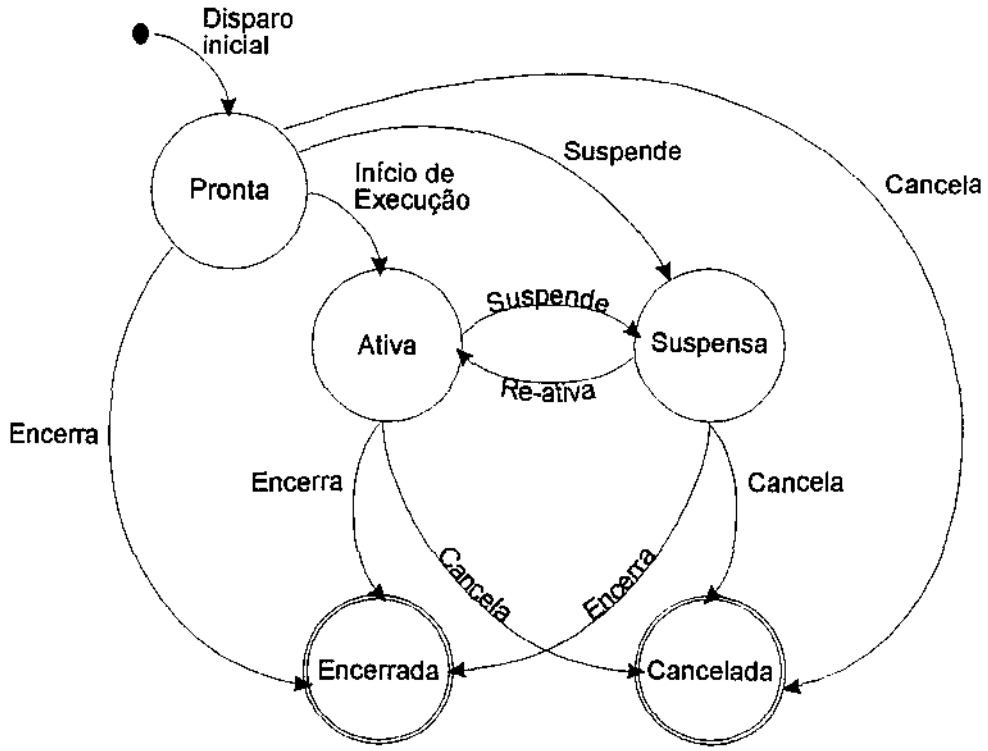


Figura 48 - Ciclo de vida de atividades.

A transição entre estes estados exibidos na figura 48 pode ser comandada pelas ações que examinaremos a seguir.

- **Encerramento(rótulo:string)**

A atividade identificada é encerrada, como se o seu final tivesse sido sinalizado (o que corresponde à geração de uma *t-action*). A representação gráfica é apresentada pela fig. 49-a.

- **Cancelamento(rótulo:string)**

A atividade identificada é abandonada, sem que seja gerado o sinal correspondente ao seu término (a *t-action* não é gerada). Em conseqüência da não geração da *t-action*, pode não ocorrer alguma habilitação posterior que depende deste sinal. É atribuição dos agentes evitar ou corrigir qualquer conflito que decorra de um cancelamento. A representação gráfica é apresentada pela fig. 49-b.

- **Suspensão(rótulo:string)**

A atividade é momentaneamente suspensa, significando que os executores serão avisados de que não devem realizar nenhum tipo de trabalho relacionado a esta atividade. A suspensão pode ser determinada para que uma situação anormal possa ser analisada e corri-

gida antes que se dê prosseguimento ao processo. A representação gráfica é apresentada pela fig. 49-c.

- **Reativação(rótulo:string)**

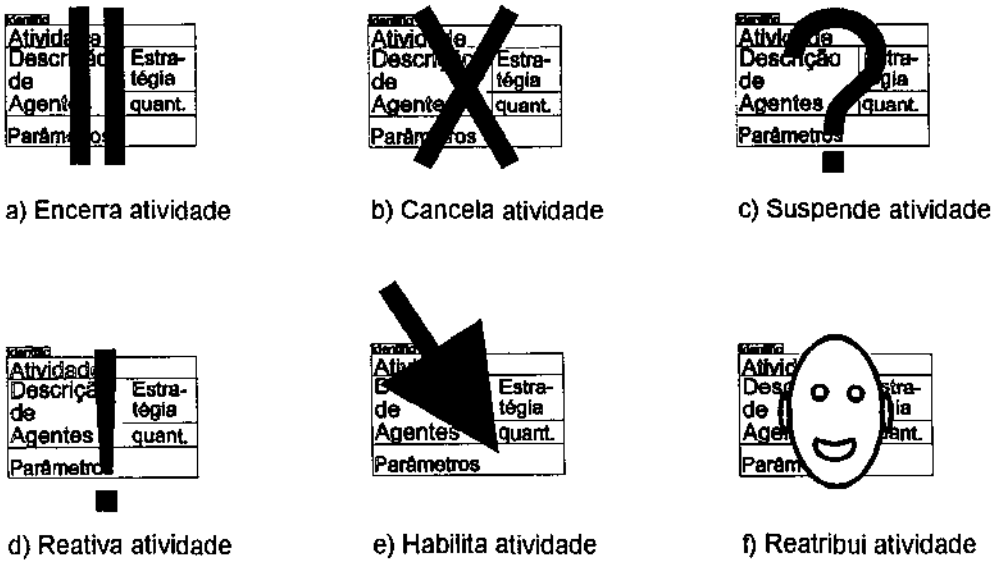
A atividade mencionada volta a estar em estado ativo ou pronta (dependendo de seu estado inicial), após a suspensão: os executores são notificados e os trabalhos liberados. A representação gráfica é apresentada pela fig. 49-d.

- **Habilitação(rótulo:string)**

Dá inicio a uma atividade qualquer do plano, forçando a habilitação das condições de disparo. Esta ação corresponde a um salto incondicional para a etapa mencionada. A execução das demais ações em curso tem continuidade, concorrentemente à nova ação habilitada, a não ser que sejam explicitamente canceladas pelos agentes. Podemos ter a situação em que a ação progride em duas frentes simultaneamente, uma no início de um *thread* e a outra do meio para adiante. Quando a atividade habilitada é terminada, a *t-action* gerada ao seu final pode vir a habilitar outras ações que a sucedem, caso as pré-condições tenham sido satisfeitas. É responsabilidade dos agentes evitar conflitos que possam decorrer desta situação. A representação gráfica é apresentada na figura 49-e.

- **Reatribuição(rótulo:string, {descr. agentes:string}, *n:número, estratégia:string*)**

A atividade identificada é retirada dos seus atuais executores e colocada sob os cuidados de outros agentes, cujas descrições são passadas. Esta ação pode ser utilizada para substituição de algum agente que esteja impedido de realizá-la por qualquer motivo (ausência no trabalho, p.ex.). A representação gráfica é apresentada pela fig. 49-f.



a) Encerra atividade b) Cancela atividade c) Suspende atividade

d) Reativa atividade e) Habilita atividade f) Reatribui atividade

Figura 49 - Representação das ações de manipulação de atividades.

Examinaremos a seguir as ações relativas à manipulação de *threads*. Estas ações oferecem funcionalidade semelhante à que acabamos de examinar, só que se aplicam a diversas atividades simultaneamente, que tem em comum o fato de pertencerem a um mesmo *thread* de processamento.

6.2.3.3. Manipulação de *threads*

Esta categoria de ações é bastante similar às ações de manipulação de atividades que acabamos de examinar, com a diferença de que atuam sobre todas as atividades correntes de um *thread* de uma só vez. Cada *thread* é identificado pelo evento que lhe dá início, como, por exemplo, o evento de disparo inicial, que dá início ao *thread* default de processamento de um plano. Outros eventos assíncronos podem também gerar seus próprios *threads*, como por exemplo, um pedido de cancelamento gerado por um cliente (veja mais detalhes em 6.2.1.-EVENTOS E THREADS).

- **Ativa thread(thread:identthread, *{parâmetros:objeto}*)**

Comanda o disparo de um evento qualquer. O plano deve ter previsto o tratamento deste evento. A existência desta ação permite que se possa comandar, a partir da própria especificação, ou de forma *ad-hoc*, o disparo de um *thread* qualquer de tratamento de evento. A notação gráfica para ativação de evento pode ser vista na figura 50-a.

Os parâmetros, opcionais, permitem que informações iniciais sejam passadas para o trecho de tratamento deste evento. Estes parâmetros devem corresponder aos especificados no evento que dá início ao *thread* correspondente (veja 6.2.1.-EVENTOS E THREADS).

- **Encerramento do thread(*thread:identthread*)**

Todo o processamento do *thread* especificado é encerrado. Cada uma das atividades em execução deste thread é terminada, como se tivesse havido uma transição para um estado final de encerramento. Caso omitido o identificador do evento inicial, assume-se que se trata do *thread* default, correspondente ao evento inicial de disparo. A representação gráfica é apresentada pela fig. 50-b.

- **Cancelamento do thread(*thread:identthread*)**

Todo o processamento do *thread* especificado é cancelado. Cada uma das atividades em execução deste thread é abortada. Caso omitido o identificador do evento inicial, assume-se que se trata do *thread* default, correspondente ao evento inicial de disparo. A representação gráfica é apresentada pela fig. 50-c.

- **Suspensão do thread(*thread:identthread*)**

Todas as atividades do *thread* são suspensas, até segunda ordem. Caso omitido o identificador do evento inicial, assume-se que se trata do *thread* default, correspondente ao evento inicial de disparo. A representação gráfica é apresentada pela fig. 50-d.

- **Reativação do thread(*thread:identthread*)**

As atividades suspensas do *thread* são reiniciadas: os executores são notificados e os trabalhos liberados. Note que esta ação precisará ser executada sob controle de um outro *thread*, que não o suspenso, ou de forma *ad-hoc*. A representação gráfica é apresentada pela fig. 50-e.

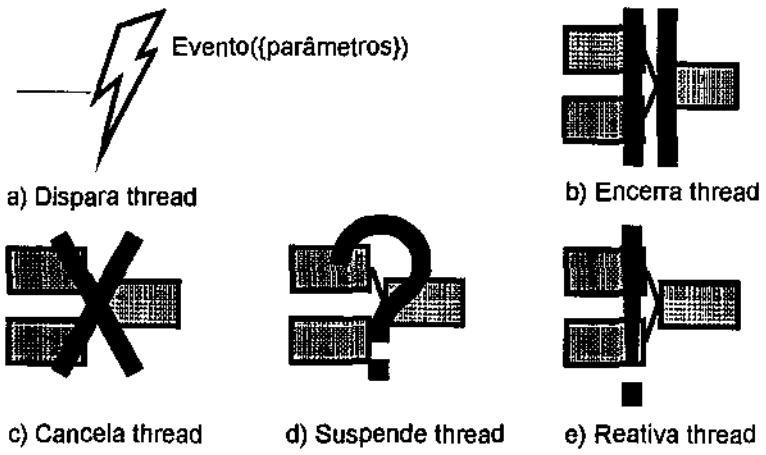


Figura 50 - Representação das ações de manipulação de *thread*.

6.2.3.4. Manipulação do caso

Além da manipulação de atividades individuais e de *threads*, podemos manipular todas as atividades correntes referentes a um caso como um todo através das ações que examinaremos a seguir. Estas ações oferecem uma maneira conveniente de se operar sobre todas as ações correntes do caso, independentemente do *thread* a que pertencem.

- **Encerramento do caso()**

O caso como um todo pode ser considerado encerrado, cessando todo o processamento, como se um marcador de fim tivesse sido alcançado. A representação gráfica é apresentada pela fig. 51-a.

- **Cancelamento do caso()**

Similar ao anterior, ocorrendo o encerramento de todo o processamento, com a diferença de que o estado de terminação é anormal, sinalizando o cancelamento. A representação gráfica é apresentada pela fig. 51-b.

- **Suspensão do caso()**

Todas as atividades do caso são suspensas, até segunda ordem. A reativação só poderá ser comandada de forma *ad-hoc*, naturalmente, já que nenhum *thread* estará mais ativo. A representação gráfica é apresentada pela fig. 51-c.

- **Reativação do caso()**

As atividades suspensas do caso são reiniciadas: os executores são notificados e os trabalhos liberados. Obviamente, esta ação só pode ser realizada de forma *ad-hoc*, já que o caso como um todo estará suspenso.

- **Troca de natureza do caso(novo plano:plano)**

Existem ocasiões em que um determinado caso muda completamente de natureza, devido a alguma contingência, como por exemplo o cancelamento de um contrato. Dependendo do caso, além de encerrar a execução das atividades do contrato, pode-se desejar iniciar um processo judicial contra o cliente. A natureza do processo se altera completamente,

sendo necessário substituir o plano por um adequado à nova situação (veja um exemplo de uso em 6.2.5.-EXEMPLOS DE PLANOS).

Uma forma menos radical de substituição acontece quando existem diversos planos alternativos para um processo e se descobre, a meio caminho, com o processo já em andamento, que existe um plano mais apropriado para tratamento do caso específico. Neste caso, o plano existente é substituído por uma variação. Representaremos esta troca de plano conforme indicado na figura 51-d.

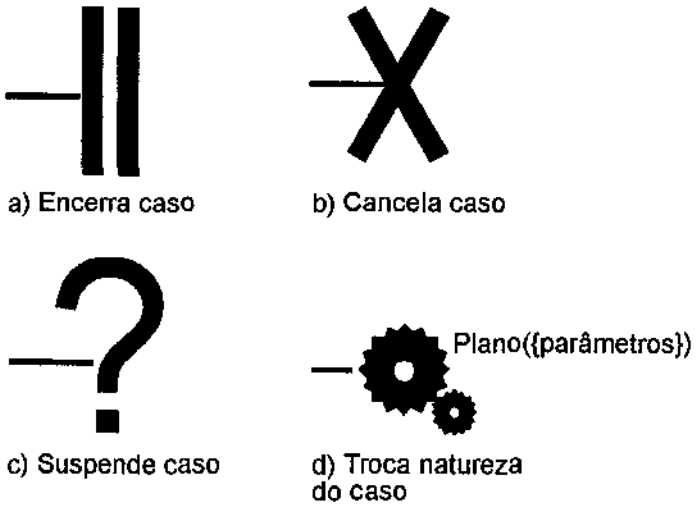


Figura 51 - Representação das ações de manipulação de dados.

6.2.3.5. Ações sobre dados

A última categoria de ações que examinaremos diz respeito à manipulação de dados. A grande maioria das discussões sobre linguagens de especificação se limita a apresentar aspectos relacionados às ações básicas de disparo de atividades. Consideramos importante que outras ações possam ser incluídas em uma especificação, de acordo com o requisito de que não haja impedância entre as ações passíveis de serem realizadas de forma *ad-hoc* e as realizadas sob controle de uma especificação. As razões disto foram já discutidas em 5.4.-AÇÕES *AD-HOC* E ESPECIFICAÇÕES.

Adiantaremos aqui estas ações, cuja funcionalidade específica conheceremos em mais detalhes quando apresentarmos a base conceitual de objetos proposta em 6.3.1.-FUNDAMENTOS PARA UM MODELO DE OBJETOS. Como teremos ocasião de verificar, os conceitos que apresentamos pressupõem uma manipulação de objetos bastante flexível, que permite, entre outras coisas, a modificação da estrutura de objetos (composição de atributos) em tempo de execução, o uso de múltiplas apresentações e assim por diante.

- **Inclui(objetodestino:identobj, componente:identobj)**

Inclui o componente especificado no objeto destino. Esta ação pode ser utilizada para adicionar uma nova informação a um objeto existente. A figura 52-a apresenta a forma gráfica desta ação, para inclusão em especificações.

- **Exclui(componente:identobj)**

Elimina o componente especificado. A representação gráfica é a da figura 52-b.

- **Cria instância(componente: identobj)**

Cria uma nova instância utilizando o componente como tipo. A representação gráfica é a da figura 52-c.

- **Inicializa(componente:identobj, atributo:identobj, valor:identobj)**

Instancia um atributo com um determinado valor, um número para um valor numérico, *string* para textual, e assim por diante. A representação é a da figura 52-d.

- **Move(componente:identobj, x:numérico, y:numérico)**

Movimenta o componente na janela de apresentação, de acordo com as coordenadas especificadas. A figura 52-e mostra a representação gráfica.

- **Apresentação(componente:identobj, apresentação:identapres)**

Para componentes que possuem várias apresentações, modifica a apresentação do componente para a especificada pelo identificador de apresentação. Esta ação é representada pela figura 52-f.

- **Método(componente:identobj, método:identmétodo, *{parâmetros:identobj}*)**

Um método é ativado, passando os parâmetros mencionados. Se o componente for um botão, por exemplo, pode-se simular o resultado de uma ativação (*click*), através da chamada do método apropriado. A notação gráfica correspondente a ativações de métodos pode ser vista em 52-g.

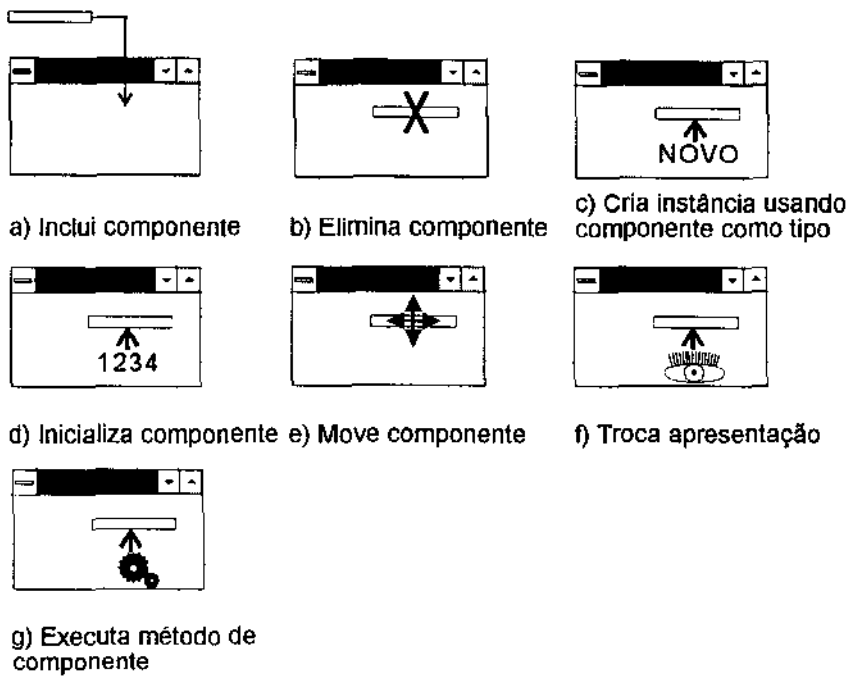


Figura 52 - Representações gráficas das ações sobre dados.

6.2.4. LITERAIS, VARIÁVEIS, MÉTODOS E PESQUISAS

Como regra geral, todo literal utilizado como parâmetro das ações e elementos de sincronismo deve poder ser substituído por uma variável, condição, chamada de método ou pesquisa sobre os dados.

Nas definições de agentes contidas nas ações de notificação e disparo de atividades, por exemplo, podemos utilizar as diversas formas:

- Literais - descrição fixa, como "João" ou "Chefe do departamento financeiro";
- Variáveis - caso as descrições se encontrem armazenadas em algum atributo ou variável. Denotaremos as variáveis em itálico, como por exemplo, *Executores*. Em qualquer caso, os tipos destas variáveis devem ser compatíveis com os parâmetros esperados na ação em que forem utilizados. O uso de variáveis permite que valores de controle sejam atribuídos diretamente pelos agentes, influenciando de forma direta a tramitação do caso;
- Métodos - pode ser utilizado o retorno de um método (desde que de tipo compatível) em substituição do literal. Denotaremos um método definido em algum outro objeto por *Objeto.Método()*;
- Pesquisas - valores resultando de pesquisas sobre bases de dados podem ser utilizados de forma similar, desde que respeitados os tipos compatíveis. O formato específico para pesquisas dependerá da linguagem de pesquisa que vier a ser utilizada.

Os elementos acima podem ser utilizados em expressões condicionais, testando-se valores de atributos, retornos de métodos e assim por diante, por exemplo (usamos uma notação informal):

```
se Valor < 10000
     Funcionário do depto financeiro
senão
     Gerente do depto financeiro
fimse
```

A condição acima determina que os agentes potencialmente responsáveis são os funcionários do departamento financeiro, caso o conteúdo de *Valor* seja inferior a dez mil, ou o gerente, caso contrário (na realidade, as descrições não podem ser tão informais como as que apresentamos, como veremos em 6.4.-SELEÇÃO DE EXECUTORES).

De forma semelhante ao exemplificado para as descrições de agentes, pode-se utilizar este tipo de expressões para suprir as demais informações relativas a ações e eventos de sincronismo.

6.2.5. EXEMPLOS DE PLANOS

A figura 54 apresenta um plano de processamento da seleção de candidatos à pós-graduação. Este é apenas um dos modos possíveis de se especificar este processo, escolhido por apresentar características interessantes do ponto de vista da linguagem de representação. Vamos examinar cada um dos elementos deste plano, rotulados no diagrama através dos pequenos retângulos com números.

1) Indicador de início de processamento. Quando uma nova instância de caso é criada, a execução se dá sempre por este elemento. Ele corresponde ao evento automático de início de caso, que dispara o *thread* default.

2) Disparo de atividade de inscrição, a ser realizada pela Secretaria de curso. Pode-se associar a esta ação o evento assíncrono opcional que estabelece um limite de data. Caso este limite seja ultrapassado, a atividade pode ser dada automaticamente como terminada, de forma a fazer com que o caso continue a fluir, mesmo que incompleto. Uma alternativa é simplesmente cancelar o caso, por falta dos subsídios obrigatórios.

3) Este batch iterativo agrupa os casos de cinco em cinco, antes de dar início à distribuição para avaliação. Como o número de candidatos provavelmente não será um múltiplo exato de cinco, o evento assíncrono associado a este conector deve liberar os casos restantes quando chegar a data limite para a inscrição, de forma que também possam ser distribuídos. O agrupamento de um número mínimo de casos permite que os agentes responsáveis pela tarefa só se preocupem com a sua realização quando um volume suficiente de casos justifique a sua intervenção. Opcionalmente, poderia ser utilizado um batch total, disparado em uma data determinada, o que na prática seria o mais normal (fig. 53).

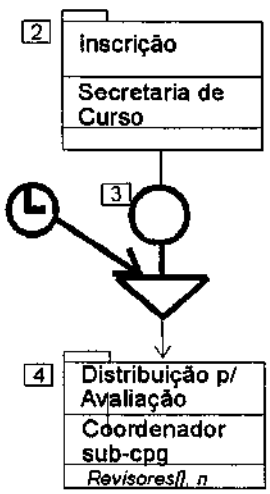


Figura 53 - Disparo baseado em data limite.

4) A distribuição para avaliação tem por objetivo escolher para cada candidato quais serão os revisores. Esta escolha pode seguir critérios subjetivos, como conhecimento específico dos revisores sobre a instituição, ou área de especialização e assim por diante. O resultado da execução desta atividade consiste na instanciação das variáveis *Revisor* e *N*, que armazenam respectivamente as descrições e a quantidade de revisores selecionados para o candidato pelo coordenador da subcomissão de pós-graduação (sub-cpg). Estas variáveis são explicitadas como parâmetros da atividade, de forma a tornar clara a sua origem.

Um evento assíncrono associado a esta ação pode enviar notificação para o coordenador da sub-cpg, quando uma determinada data limite for alcançada.

5) Este *fork* replica a próxima atividade de forma que exista uma para cada um dos *N* revisores selecionados para o candidato. Note que o fato de ser baseado em uma variável permite que seja criado um número variável de atividades replicadas, de acordo com critérios determinados pelo agente durante a execução da atividade de distribuição para revi-

são. Desta forma, cada candidato pode ser avaliado por um número diferente de executores, dependendo de condições especiais subjetivas.

6) Esta ação, replicada pelo *fork* 5 que a precede, dispara uma atividade de avaliação para cada um dos revisores cuja descrição se encontra na variável *Revisor*, um vetor instanciado na atividade disparada pela ação 4. A variável *Iter* é fornecida pelo sistema e corresponde ao número seqüencial de replicação, i.é, vale 1 para a primeira replicação, dois para a segunda e assim por diante.

7) Este *join* aguarda até que um determinado número de revisores tenha terminado a sua análise do candidato. Este quorum pode ser menor ou igual a *N*, o número total de revisores do caso, permitindo que algumas das revisões redundantes sejam ignoradas. Se desejado, pode-se utilizar um teste condicional para computar qual o quorum mínimo, baseado no tipo de inscrição do candidato, por exemplo:

```
se Tipo = "Doutorado"
     p = 3
senão
     p = 2
fimse
```

A condição anterior estabelece que, se o candidato se inscreveu no programa de doutoramento, então pelo menos três dos revisores precisam terminar a avaliação antes de que o caso prossiga, enquanto que apenas duas avaliações são suficientes no caso de outros candidatos (ao mestrado, p.ex.).

Associado a este elemento, pode-se especificar o evento assíncrono que dê a espera por encerrada quando uma data limite for alcançada, enviando o caso para a classificação independentemente do número de avaliações já realizadas. Um alternativa é simplesmente notificar (a sub-cpg, p.ex.) de que existe o atraso e aguardar a intervenção manual do responsável.

8) Este batch sincroniza todos os casos, de forma que a próxima atividade possa ter acesso a todos eles para realizar uma análise comparativa. Só se pode estabelecer a classificação final a partir da comparação de todos os candidatos, por isto obrigatoriamente deve haver o sincronismo total entre os casos.

9) O resultado final da atividade de classificação é a determinação do aproveitamento ou não do candidato no programa, registrado na variável *Result*. Esta determinação é feita com base na comparação das avaliações dos candidatos entre si.

10) Este *fork* condicional executa a ação de aceitação ou rejeição dependendo do valor da variável *Result*, instanciada na atividade 9.

11) Caso o resultado seja de rejeição, uma carta é emitida. O executor **auto** significa que a atividade é automática. Uma alternativa seria utilizar uma única atividade que gere a comunicação apropriada a partir do exame da variável *Result*.

12) Semelhante a 11, gera carta de aprovação.

13) Este marcador sinaliza o encerramento do processamento do caso.

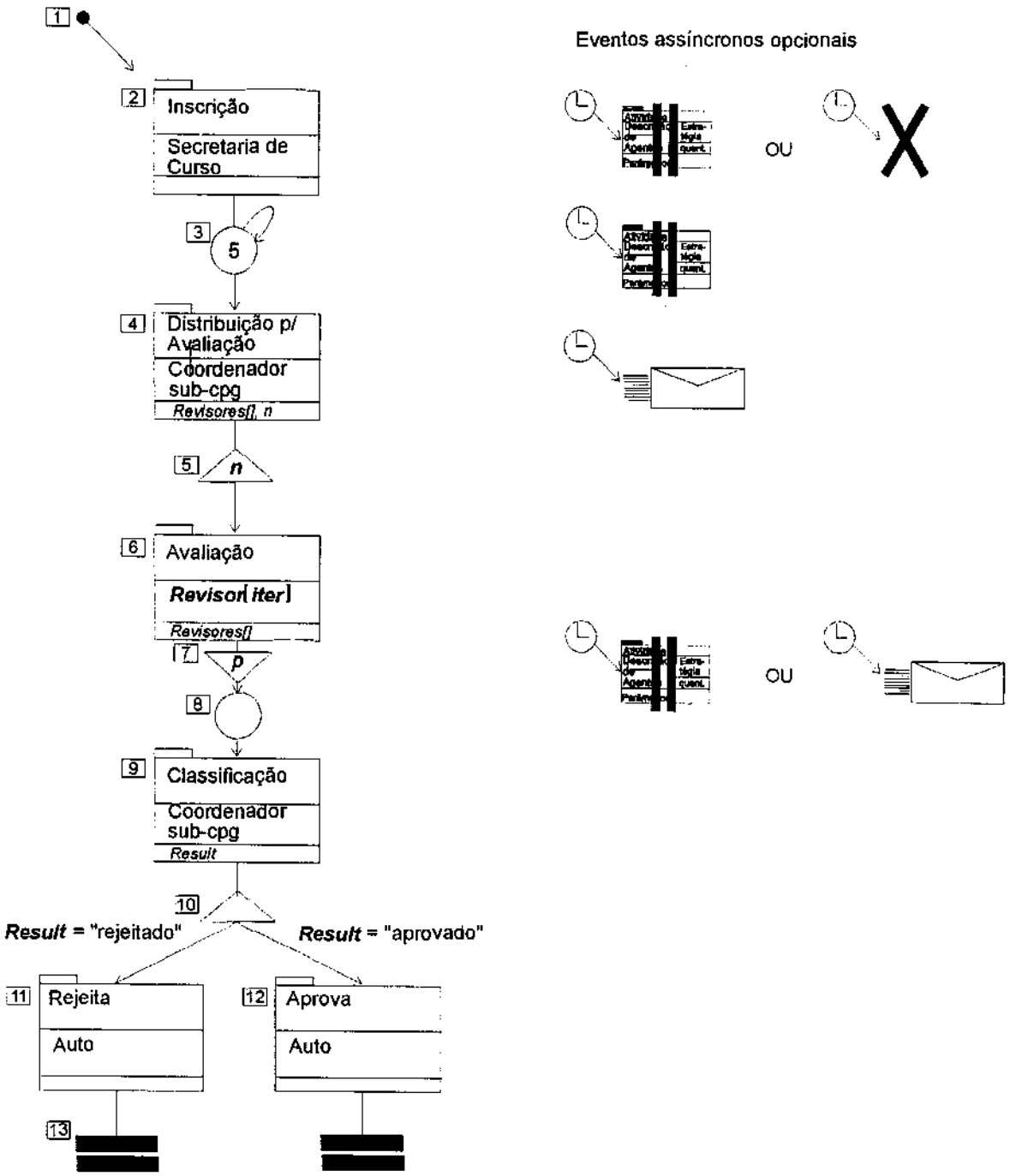


Figura 54 - Processo de seleção de candidatos à pós-graduação.

Vamos passar a examinar um outro exemplo, o de Processamento de Pedidos, cuja descrição genérica foi apresentada no início do presente texto, explorando a possibilidade de especificação de *threads* assíncronos.

O diagrama da figura 55 apresenta uma especificação que utiliza dois *threads*, o default e um de tratamento de cancelamento. O processamento do pedido em si é extremamente simples. Após o registro do pedido (1), o crédito do cliente é analisado (2). Caso seja re-

jeitado, o caso se encerra. Se aprovado, existe o faturamento e a expedição em atividades paralelas (5 e 6).

A parte mais interessante do processamento acontece no *thread* de tratamento de solicitação de cancelamento (7). Este evento dá início a um *thread* próprio, que inicia cancelando as atividades do *thread* default (8), solicitando em seguida um parecer jurídico sobre a situação, que determinará a continuidade do caso. Se o cancelamento for considerado procedente, o caso é encerrado normalmente. Caso contrário, o caso muda de natureza, havendo a troca do plano para um que conduz um processo judicial contra o cliente, através da ação 11.

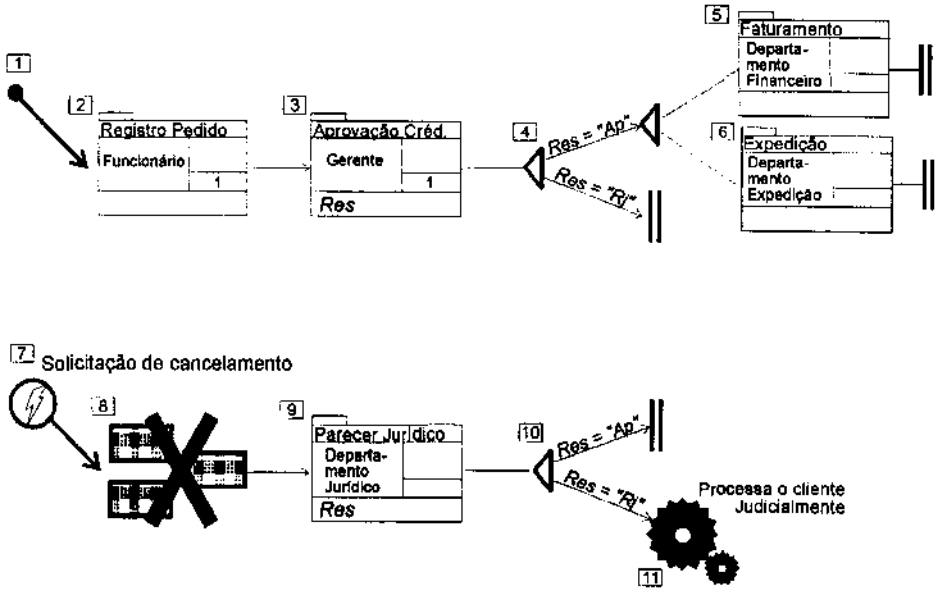


Figura 55 - processamento de pedidos.

O ANEXO 1 apresenta mais alguns exemplos de planos de processamento, referentes aos processos apresentados em 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO.

6.3. AMBIENTE DE EXECUÇÃO

Atacamos nas seções anteriores a questão do aumento de expressividade do modelo, através do tratamento de eventos, elementos de sincronismo e ações, normalmente não oferecidos pela maioria dos modelos. Isto por si só não é suficiente para alcançarmos os objetivos de fácil utilização que propusemos no capítulo 5.

Discutiremos agora uma parte complementar, igualmente importante, referente ao ambiente de execução. Apesar da maioria das discussões encontradas na literatura centrarem apenas na expressividade da linguagem, julgamos que existem dificuldades de uso que só podem ser resolvidas através da oferta de um ambiente de execução igualmente poderoso.

Vimos em 3.4.-AÇÕES AD-HOC E ESPECIFICAÇÕES, que na presença de contingências, por mais poderosa que seja uma especificação, sempre existirão ocasiões em que outros tipos de ações precisarão ser executadas, de forma *ad-hoc*. Esta liberdade de ação, que pode ser associada à corrente interpretivista, é omitida na maioria dos sistemas, causando dificuldades no tratamento de exceções.

Como vimos em 3.5.-VISÕES DE MUNDO, podemos associar as ações *ad-hoc* à existência de aspectos guardião e comunicador mais fortes, justamente os aspectos menos enfatizados na maioria dos sistemas. Para cumprirmos o requisito de suporte a ações *ad-hoc*, vamos, portanto, estabelecer as bases conceituais de um modelo de objetos que se adeqüe aos requisitos de flexibilidade fundamentais em sistemas de workflow. Vamos relembrar aqui algumas das questões principais referentes a esta flexibilidade:

- **Modelagem** - as observações, principalmente de autores ligados à etnografia ([Rob93], [RB91], [Suc87], [Sac95], [Kyn95], p.ex.), indicam que a modelagem realizada por pessoal diferente do diretamente envolvido no trabalho em si pode apresentar graves distorções entre o modelo e a realidade do trabalho. É primordial, portanto, que o usuário final possa se envolver diretamente neste trabalho de modelagem, sem intermediários;
- **Adaptação a circunstâncias especiais** - as exceções ocorrem em momentos imprevistos e precisam ser tratadas pelos próprios usuários: não há tempo para que uma equipe técnica seja acionada a cada ocorrência excepcional. O volume de ocorrência de exceções é muito maior do que se poderia imaginar, mesmo em processo "bem comportados", com alto grau de estruturação. A existência das exceções torna necessária a adaptação de casos de forma não antecipada, através da adaptação dinâmica dos casos, o que torna inadequado o paradigma tradicional de desenvolvimento, que separa claramente a modelagem da execução: o planejamento e a execução são necessariamente entremeados;
- **Evolução** - a realidade externa que se modela não é estática, mas se encontra em constante mutação [EKR95, Saa94]. Esta modificação da realidade tem que se refletir nos modelos utilizados, de forma a manter a sintonia do sistema com esta realidade. Note que a evolução do modelo afeta tanto os casos futuros, que virão a ser criados, como pode afetar também casos já em andamento, correspondendo neste último caso à evolução dinâmica (*"dynamic change"* [EKR95]).
- **Suporte ao individualismo** - não existe um único modo de se atingir um objetivo de negócio: diversos processos variantes são aplicados, dependendo do caso, da unidade da organização ou do agente que está realizando as tarefas;
- **Acesso não antecipado a informações** - diversos agentes precisam ter acesso simultâneo às informações dos casos em tramitação, por diversos motivos. Alguns deles executarão algumas das atividades de cada caso, por solicitação do sistema. Outros necessitam observar os casos para se manterem informados de seu andamento. Os observadores não tem um papel ativo na maior parte do tempo, não executando nenhuma das tarefas específicas relacionadas a eles. Quando surge uma contingência, porém, estes observadores se transformam repentinamente em executores, agindo de forma a corrigir a anormalidade. O conhecimento acumulado pela observação da tramitação precedente é utilizado neste momento como subsídio para a correta tomada de decisão, sendo, portanto, fundamental.

Note que as questões que enunciamos acima se aplicam tanto à modelagem de **objetos de aplicação**, quanto aos **objetos de sistema**:

- Objetos de aplicação são aqueles que armazenam as informações sobre cada pedido, candidato, mercadoria, por exemplo. Eles contém as informações referentes à realidade externa, e constituem o produto visível do trabalho realizado pelos agentes em relação a cada caso;
- Objetos de sistema armazenam as especificações de processo, como o plano de processamento, o modelo organizacional e assim por diante. Estes objetos contém as meta-informações, que são interpretadas pelo sistema e que determinam, em última instância, a tramitação de cada caso.

Ambos os tipos de objetos estão sujeitos a modificações e adaptações durante a tramitação de um caso. Os dados de um candidato à seleção à pós-graduação, por exemplo, precisam ser modelados, adaptados, evoluem e precisam se adaptar a peculiaridades dos casos e dos agentes. O mesmo acontece com informações deste processo: as etapas de processamento da seleção da pós-graduação precisam ser definidas, adaptadas em casos especiais, evoluem e precisam se adaptar a peculiaridades dos casos e dos agentes.

Note que a perspectiva implícita nos parágrafos anteriores é inovadora em dois aspectos, em relação ao que é tradicional na área de workflow: 1) o suporte à modelagem, evolução, adaptação, etc. dos objetos de aplicação raramente é tratado, sendo normalmente considerada externa ao escopo do sistema; 2) as informações de sistema não costumam ser vistas como objetos semelhantes aos de aplicação, mas são zelosamente guardadas sob controle do sistema, exigindo um modo especial de manipulação, que envolve eventualmente até a recompilação [AS94].

O que propomos é a utilização de um substrato único de gerenciamento de objetos, que propicie a funcionalidade necessária à manipulação homogênea dos dois tipos de objetos, de forma indistinta. A vantagem deste tratamento é que as mesmas ferramentas são utilizadas na manipulação de qualquer objeto, o que, além de facilitar o aprendizado dos usuários, permite que um requisito primordial possa ser atendido, o tratamento homogêneo da modelagem e execução: ambos passam a ser realizados através da manipulação de objetos (de sistema e aplicação, respectivamente), feitas da mesma maneira.

Um modelo de objetos que vise dar suporte a sistemas de workflow que atendam aos requisitos expostos deve necessariamente estar preparado para enfrentar a fluidez excepcional que estes requisitos impõem, o que não acontece, infelizmente, com nenhum modelo de dados de que tenhamos conhecimento.

Não nos resta senão a alternativa de propormos a nossa própria base conceitual, na expectativa de que um modelo adequado venha a ser desenvolvido no futuro. Os conceitos que passamos a apresentar farão uso intensivo de uma série de recursos da área de Bancos de Dados, muitos dos quais correspondem a áreas de pesquisa não completamente equacionadas.

Apresentaremos inicialmente a discussão sobre os fundamentos deste modelo, exemplificado através da manipulação de objetos de aplicação (6.3.1.-FUNDAMENTOS PARA UM MODELO DE OBJETOS). Em 6.3.2.-CONTEXTOS DE EXECUÇÃO, mostraremos que os mesmos conceitos podem ser utilizados também para a manipulação de objetos de sistema.

6.3.1. FUNDAMENTOS PARA UM MODELO DE OBJETOS

Apresentaremos a base conceitual de um modelo que sirva de substrato para a construção de objetos cuja manipulação atenda aos requisitos de flexibilidade e adaptabilidade necessários. Não estamos preocupados neste momento com questões de eficiência e implementação (aliás bastante árduas).

Baseado no fato de que não é possível se antecipar as necessidades de informações e a sua evolução, um dos conceitos fundamentais é o de se permitir que o próprio usuário disponha de ferramentas que permitam a definição e evolução de esquema, além daquelas de consulta e atualização que normalmente são oferecidas por qualquer sistema flexível.

Tendo isto em vista, propomos que a representação de dados e meta-dados seja feita de uma forma unificada, formando uma hierarquia diretamente manipulável pelos usuários. Nosso objetivo é o de tornar a barreira que separa tipos de instâncias a mais tênue possível, de forma a poder permitir a manipulação de classes e objetos de uma maneira direta e uniforme.

O uso mais comum do mecanismo de classificação em gerenciamento de dados é o da modelagem da dicotomia tipo básico / instância, o mecanismo de abstração entre o esquema e a estensão, que ocasiona uma divisão clara entre tipos, geralmente embutidos nas aplicações, e as instâncias, manipuladas pelos usuários. Pirotte et al. propõem em [PZM+94] um mecanismo de materialização no qual são definidas uma série de classes crescentemente mais concretas, do qual o tipo básico e as instâncias podem ser vistos como extremos de um espectro contínuo, com a classe (com faceta de objeto nula) em um extremo e as instâncias (materializadas) em outro.

As classes, neste modelo, possuem duas facetas, a de **classe** propriamente dita, que corresponde à especificação de atributos e seus tipos, como é habitual, e a faceta de **objeto**, em que alguns destes atributos recebem valores. Podemos, por exemplo, definir uma classe automóveis, com atributos de marca, modelo e cor e as sub-classes automóveis-azuis e automóveis-vermelhos, que possuem uma faceta de objeto onde o atributo de cor possui valores específicos (azul e vermelho, respectivamente). A faceta de objetos fornece valores aos atributos que podem ser encarados como constantes aplicáveis sobre as instâncias: todas as instâncias da classe automóveis-vermelhos terão cor vermelha, naturalmente.

Note que se levarmos o mesmo raciocínio adiante, podemos enxergar as instâncias como sub-classes nas quais a faceta de objeto possui valores para todos os atributos (ou pelo menos para o conjunto de atributos que garante a identidade do objeto). É justamente este aspecto que nos atrai na proposta de Pirotte.

Do modelo de Pirotte et al., nos apropriaremos de duas propostas: 1) o tratamento de meta-dados como dados, propiciando em princípio a manipulação uniforme entre tipo e instância e 2) o fato de que as classes neste modelo possuem uma faceta de objeto, em adição à faceta de classe que é habitual. Vamos justificar a necessidade para estas duas características a seguir:

1) Em sistemas de workflow, o papel reservado normalmente aos analistas e programadores passa a ser executado pelos próprios usuários, por força da necessidade de adaptação de casos para se fazer frente a contingências. Os meta-dados ou tipos precisam, portanto, estar acessíveis para manipulação pelos usuários, preferencialmente sem que seja

necessária a utilização de ferramentas especiais. O paradigma das planilhas de cálculo, que já apontamos como sendo mais apropriado, nos inspira a evitar que exista um modo especial de modelagem, distinto do modo de execução. Esta posição é corroborada por observações de autores como Malone et al. [MLF92].

2) O fato de que neste modelo classes são mais do que receptáculos vazios, podendo armazenar informações, vai nos permitir utilizá-las para representar os objetos de sistema, como os planos de processamento que, como vimos, não são meras cascas vazias, correspondendo a estruturas de dados carregadas de informações. Poderíamos ter optado por um mecanismo diverso, com efeito semelhante, tratando os objetos de sistema como algo de especial, diferenciado dos objetos de aplicação. O que nos atrai na materialização é justamente a uniformidade de tratamento que o conceito propicia, não nos obrigando a fazer distinções entre os objetos de aplicação e de sistema, que passam a ter os mesmos privilégios e são manipulados da mesma maneira.

A figura 56 apresenta o diagrama que utilizaremos para representar hierarquias de objetos. Nesta árvore, as classes mais genéricas correspondem aos nós com menor profundidade. As sub-classes são representadas pelos nós filhos, enquanto que as folhas correspondem a instâncias. As linhas pontilhadas significam que a hierarquia pode ser estendida pelo acréscimo de outro objetos nos níveis correspondentes. O **objeto raiz** é o ancestral comum a todos os objetos existentes.

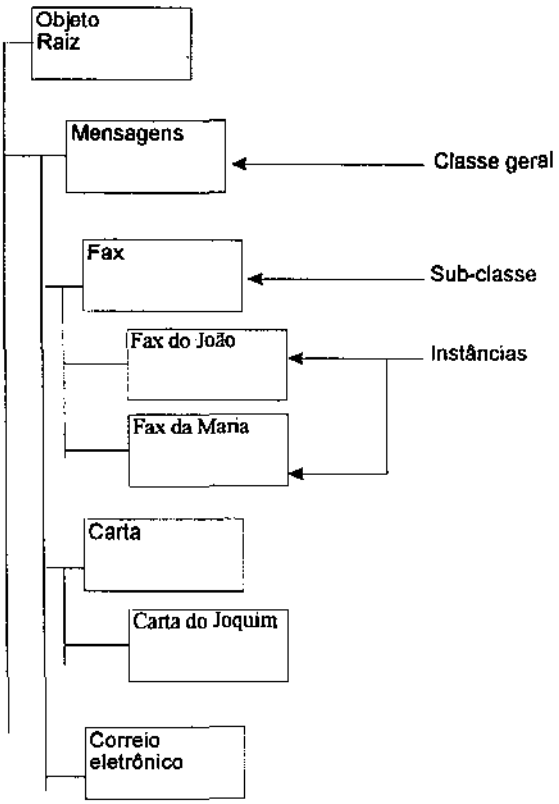


Figura 56 - Hierarquia de objetos.

A figura 56 apresenta uma classe de mensagens, que possui as sub-classes fax, carta e correio eletrônico, que por sua vez possuem instâncias específicas (o fax do João, da Maria, etc.). Utilizaremos a notação da figura 56 para representar tanto o relacionamento **é-um** decorrente da especialização por herança, quanto o decorrente da especialização via maior "concretude", correspondente à materialização, baseada em predicados, ou seja, baseada em uma faceta de objeto com maior quantidade de atributos instanciados [PZM+94].

Qualquer objeto da hierarquia pode ser utilizado em dois contextos, sendo utilizado como **tipo**, a partir do qual instâncias podem ser criadas ou como **objeto** propriamente dito, armazenando informações. Utilizaremos os termos **tipo** ou **classe** para nos referirmos ao primeiro modo e **instância** para o segundo. É importante salientar que todos estes termos dizem respeito a objetos diretamente manipuláveis pelos usuários. Utilizaremos um termo ou outro dependendo da função em que objeto que estiver sendo empregado a cada momento, e não baseados em alguma característica exclusiva que separa tipos de instâncias.

Agora que temos uma hierarquia diretamente manipulável, onde objetos e tipos convivem, e são até indistingüíveis, que espécie de utilização daremos a eles? O uso das instâncias é imediato e óbvio: instâncias armazenam informações como as referentes ao fax da Maria, ou à carta do João. Quanto aos objetos como o "Mensagens", que corresponde a uma generalização, podemos utilizá-lo de diversas formas:

1) Como tipo - a partir de qualquer objeto, podemos criar novos objetos, novas instâncias que utilizam este objeto como modelo. Para isto, todo objeto possui um método de criação de instâncias, que aloca um novo objeto idêntico (um clone), inserindo-o na coleção de objetos subordinados ("filhos");

2) Para armazenar valores comuns às instâncias - caso existam valores de atributos que se repitam nas instâncias subordinadas, estes valores podem ser fatorados das instâncias, sendo armazenados apenas no objeto que serve de tipo comum a elas. Caso todas as mensagens de fax tenham um código em comum, por exemplo, o objeto "Fax" pode armazenar este código, que será compartilhado por todas as instâncias subordinadas a ele, por todos os faxs específicos, para os quais este código funciona como uma constante. Este aspecto corresponde à materialização proposta por [PZM+94];

3) Como mecanismo de acesso - a partir de uma generalização, podemos ter acesso às instâncias a ele subordinadas, como se seus filhos não passassem de elementos armazenados em uma lista embutida no próprio objeto (o que incidentalmente pode corresponder a uma opção de implementação do mecanismo). A partir do objeto "Mensagens", por exemplo, temos acesso a seu subordinados, "Fax", "Cartas" e "Correio eletrônico". A partir de "Fax", temos, por sua vez, acesso ao "Fax da Maria" e ao "Fax do João" e assim por diante.

Voltaremos a abordar estas características em breve, quando as aplicarmos para a criação de instâncias em 6.3.1.2.-CRIAÇÃO DE NOVAS INSTÂNCIAS.

Começaremos a apresentação dos conceitos, situando-os em um cenário de uso [Kyn95] de objetos de aplicação (seção 6.3.1.1). Em seguida, vamos aplicar os mesmos conceitos na construção de objetos de sistema (seção 6.3.2). Este cenário apresentará a criação de tipos, de instâncias, a adaptação e evolução e o acesso a objetos construídos de acordo

com os conceitos apresentados, o que deve nos dar uma boa idéia sobre a adequação do modelo ao ambiente flexível de execução que desejamos.

6.3.1.1. Criação de novos tipos

A construção de novos objetos e sua adaptação deve ser feita através de métodos já conhecidos dos usuários, que estão acostumados a interagir com os sistemas através da manipulação de elementos de interface. A cultura adquirida por estes usuários na utilização de interfaces gráficas de usuário, oferecidas pela maioria dos sistemas operacionais (MacOS, Windows, Unix, p.ex.) pode ser aproveitada na construção de uma ferramenta que utilize estes elementos para a construção e modificação de estrutura dos objetos por manipulação direta [Oli93].

Uma fonte de inspiração para uma ferramenta deste tipo são os programas que permitem a criação de apresentações gráficas, como o Visual Basic, Delphi, e diversos outros editores de recursos que acompanham ferramentas de desenvolvimento para plataformas que pressupõem o uso de uma interface gráfica de usuário. Nestas ferramentas, o programador adiciona elementos de interface, como botões e caixas de edição (*edit boxes*), a partir de um conjunto de elementos básicos (*widgets*) disponível para esta finalidade, dispondo-os onde deseja através de ações realizadas com o *mouse*, como o arrasto, por exemplo. De forma resumida, estes programas pressupõem o seguinte:

- Uma janela de fundo, onde os elementos são posicionados;
- Um conjunto de elementos básicos a partir da qual os componentes da apresentação são escolhidos;
- Representações gráficas para cada um destes elementos, que podem ser manipulados através do *mouse*;

O que desejamos é permitir a construção de novos objetos a partir destes mesmos fundamentos. Note que nosso objetivo é mais abrangente do que simplesmente o de definir uma apresentação: queremos simultaneamente determinar a posição dos novos objetos na hierarquia de objetos existentes, bem como a sua composição (atributos).

A partir do objeto raiz, ancestral comum a todos os objetos, e um conjunto de tipos básicos, como campos textuais e valores numéricos, por exemplo, queremos poder construir incrementalmente novos tipos de objetos complexos, via herança e composição. O que faremos é associar cada tipo básico a um objeto próprio, que possui uma ou mais apresentações gráficas (fig. 57).

O objeto raiz e os novos objetos que viermos a criar possuirão uma janela de diálogo, através da qual incluiremos, excluiremos e re-arranjaremos os componentes. Em especial, o objeto raiz possui uma janela de diálogo vazia, sem nenhum componente.

Como qualquer objeto deve poder ser utilizado também como tipo, a partir do qual novos objetos subordinados podem ser criados, vamos determinar que todos possuam um método de criação de instância que tem a seguinte função: ao acionarmos este método de um dos objetos, uma cópia exata do objeto (clone) é criada, e este novo objeto é incluído na coleção de objetos subordinados ao original, ao mesmo tempo criando o novo objeto utilizando o original como modelo e estabelecendo um relacionamento **é-um** entre a cópia e o original.

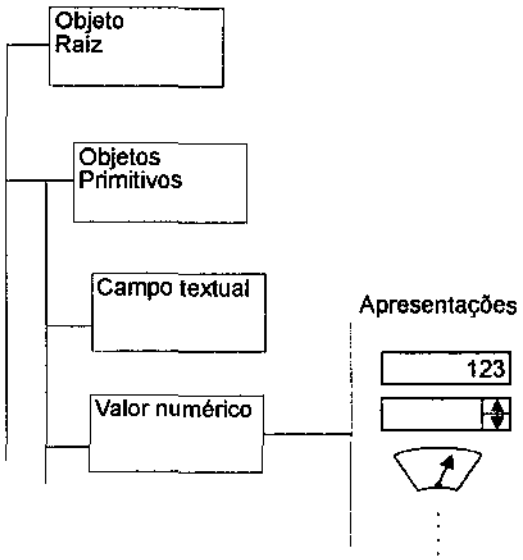


Figura 57 - Elementos básicos de construção.

Através da janela de diálogo do novo objeto, a princípio idêntica à do objeto que serviu de tipo, aplicamos as especializações, que podem corresponder a instanciações de componentes existentes, remoção ou adição de componentes:

- A instanciação de componentes corresponde à materialização [PZM+94].
- A adição de componentes corresponde à adição de atributos em uma sub-classe;
- A remoção de componentes corresponde à herança seletiva ("*selective inheritance*" [EN94]);

Os componentes adicionados podem ser quaisquer objetos existentes, os correspondentes aos tipos primitivos, e mesmo os objetos complexos já existentes. Para facilitar a manipulação destes objetos complexos, admitiremos que cada um deles também possa ser representado por um ícone, em adição à janela de diálogo que todos devem possuir.

Antes de examinarmos um exemplo, vamos ver qual a correlação entre o mecanismo proposto e as ferramentas de construção de interface que mencionamos no início da seção. O manuseio dos componentes se dá de forma semelhante, por arrasto da representação gráfica de elementos básicos sobre uma janela de definição. Ao contrário dos editores de apresentações, porém, novos objetos são construídos por herança a partir de objetos existentes e podem incluir por sua vez componentes complexos.

Vamos ilustrar o mecanismo proposto através de um exemplo, o da construção de um novo tipo de objeto que nos permita a armazenagem de dados dos candidatos à seleção à pós-graduação. Discutiremos mais adiante a adaptação deste mesmo objeto, sua evolução, e adaptação a peculiaridades de agentes, onde realmente poremos à prova a flexibilidade dos conceitos propostos.

Um candidato possui um série de informações pessoais, como nome e endereço, por exemplo, e um currículo. O currículo pode ser recebido via fax, correio eletrônico e carta comum. Não se pode determinar a *priori* qual tipo de mensagem será recebida em cada caso, e nem o momento de sua chegada.

Para criar um novo objeto que represente os candidatos, iniciamos a especialização a partir do objeto raiz, criando uma nova instância a partir dele. Este objeto raiz não possui componentes e sua apresentação corresponde a uma janela de diálogo vazia. Os componentes básicos são selecionados a partir da hierarquia de objetos disponíveis e dispostos na janela correspondente à apresentação do novo elemento (via arrasto, p.ex.). Ao definirmos a interface, definimos implicitamente os componentes deste novo objeto. As informações pessoais são representadas por componentes primitivos, como campos textuais relativos a nome e endereço. O currículo corresponderá a um componente do objeto "Mensagens", que é uma classe que engloba todos os tipos mencionados de comunicação, fax, correio eletrônico e carta (fig. 57).

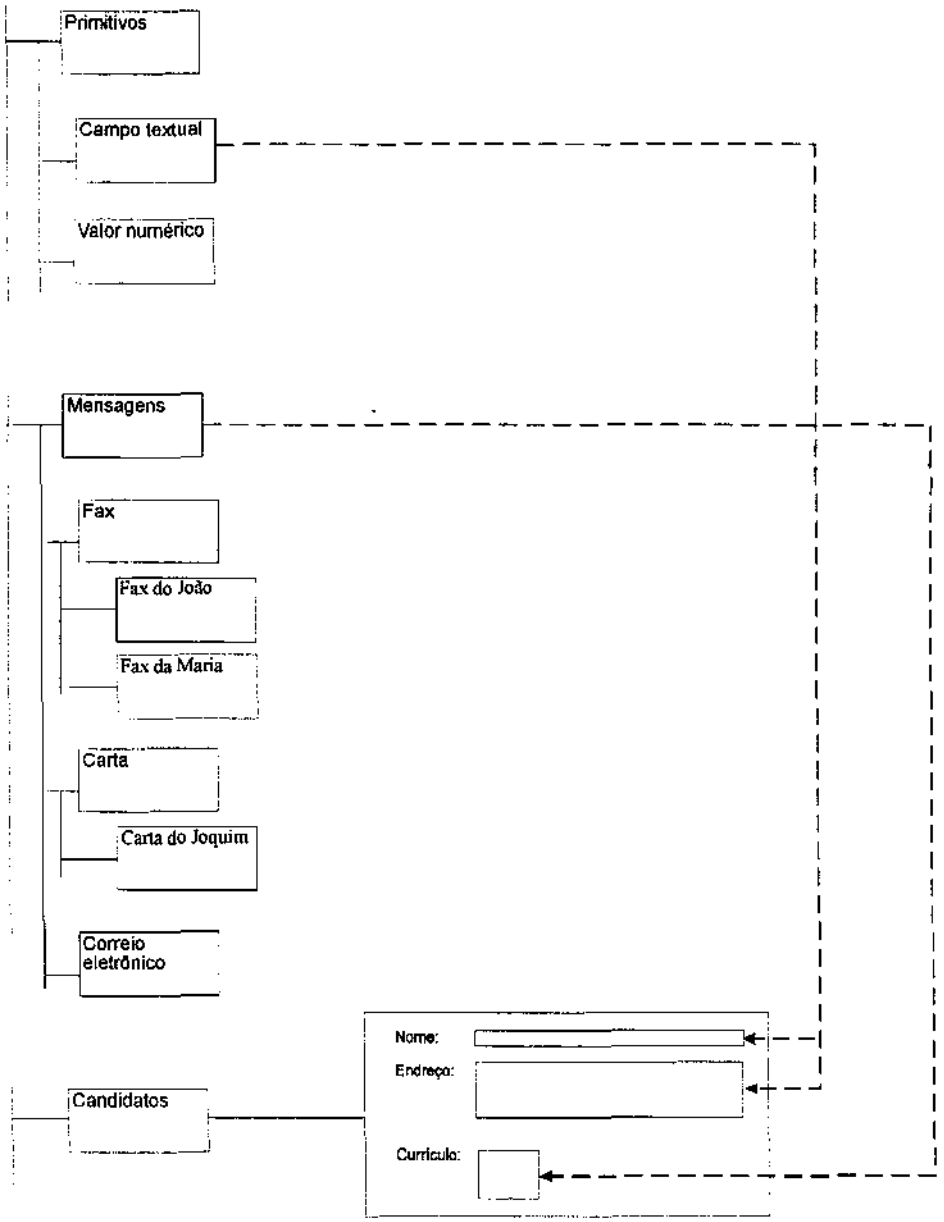


Figura 57 - Construção do objeto "Candidatos".

Note que o componente currículo corresponde a uma generalização, a de "Mensagens". Isto traz uma série de conseqüências de uso no momento em que criarmos novos objetos a partir deste tipo, que passamos a apresentar a seguir.

6.3.1.2. Criação de novas instâncias

Acabamos de ver como é feita a criação de objetos que representam tipos. Vamos examinar agora como podemos criar e utilizar instâncias a partir de um tipo como o representado pelo objeto "Candidatos" que acabamos de construir. Veremos que a criação de instâncias é feita de forma bastante similar à que já examinamos: a partir do objeto que desejamos utilizar como tipo ("Candidatos"), acionamos a opção de criação de instância, o que, como já sabemos, ocasiona a criação de um novo objeto (clone), subordinado ao tipo.

No caso de uma instância, a especialização da réplica criada corresponderá principalmente à instanciação dos atributos do objeto com valores específicos, como o nome e o endereço do candidato cujos dados se deseja armazenar. Como estes atributos correspondem a objetos primitivos, a campos textuais, os valores são informados através da digitação nos seus respectivos elementos de interface (fig. 58).

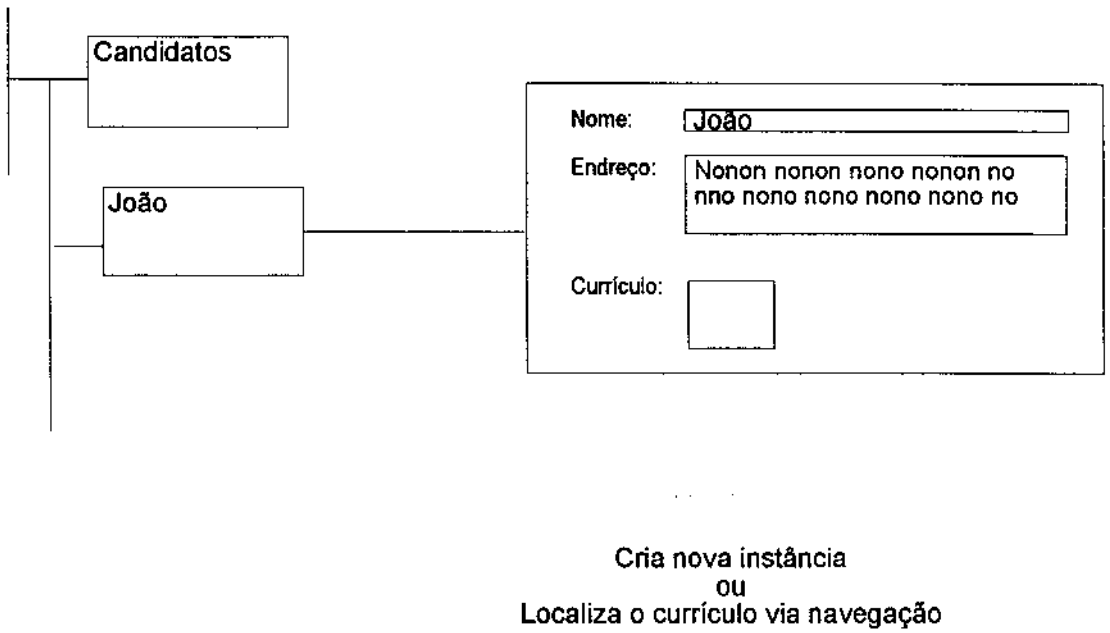


Figura 58 - Instanciação de atributos do candidato João.

Chegamos a um momento importante em nossa apresentação, o uso do componente Currículo, cujo tipo é uma generalização ("Mensagens"). Vamos iniciar a discussão deste aspecto a partir da necessidade, para então discutir a funcionalidade proposta. Que espécie de necessidade temos ao manipularmos informações sobre currículos?

- Quando um currículo é recebido, podemos querer criar um novo objeto para registrar as informações desta mensagem, que é então associada ao objeto do candidato a que diz respeito;
- A recepção de documentos pode ser executada em um outro setor, de recepção, por exemplo, que se encarrega de registrar todas as mensagens de acordo com o seu tipo (fax, correio eletrônico, carta). Neste caso, desejamos localizar o objeto correto, associando-o ao objeto do candidato;

As necessidades são, portanto, de criação de novos objetos do tipo "Mensagens" ou de busca de um dos objetos subordinados existentes. São justamente estas as funcionalidades que associaremos a componentes deste tipo, de uma maneira bastante simples. Vimos que qualquer objeto pode ser utilizado como tipo a partir do qual objetos subordinados podem ser criados. Foi justamente isto que fizemos para criar tanto o objeto "Candidatos" quanto o objeto que armazena as informações do candidato "João". A referência ao objeto "Mensagens", que incluímos via arrasto, pode ser utilizada da mesma maneira, para a criação de nova mensagem, resolvendo a primeira parte do problema. Em resumo: referências a objetos tem a mesma funcionalidade que os objetos originais, entre elas a capacidade de se criar novas instâncias baseadas nelas.

A segunda parte, a localização de um objeto já existente, também pode ser resolvida de maneira simples. Um sistema que implemente o mecanismo proposto necessariamente precisará manter, para cada generalização, a lista dos objetos subordinados a ele, o registro do relacionamento ***is-a*** estabelecido durante a ativação do método de criação de instância. Ora, como os elementos subordinados são conhecidos, basta que o sistema apresente estes objetos de uma maneira conveniente, que permita a busca. O sistema Oval [MLF92] nos oferece uma fonte de inspiração: neste sistema, grupos de objetos podem ser examinados através de diversas apresentações padrão pré-fornecidas, como tabelas, calendários e árvore, por exemplo.

Adotaremos uma solução semelhante, permitindo que a partir de um objeto ou referência a um objeto que corresponda a uma generalização se possa navegar pelos objetos subordinados utilizando-se uma das apresentações padrão do sistema, a ser escolhida livremente pelo usuário no momento que este desejar localizar alguma informação.

A partir do momento que uma nova instância de mensagem é criada, ou que uma mensagem existente é localizada e selecionada através da navegação, a referência correspondente ao currículo deve passar a apontar para esta mensagem mais específica. A partir daí, o componente currículo dará acesso diretamente à mensagem apropriada, e não mais a uma mensagem genérica.

Podemos resumir a funcionalidade de componentes que são referências a generalizações:

- Podem ser utilizados como tipo, para criação de novas instâncias;
- Permitem o acesso aos objetos subordinados, através de apresentações padrão fornecidas pelo sistema (tabelas, árvores, etc.);
- Podem passar a referenciar objetos mais específicos. Uma referência permite, portanto, um uso polimórfico, ou seja, referências a classes de mais alto nível hierárquico podem ser substituidas por referências a sub-objetos de grau hierárquico mais baixo (mais concretos, nos termos de [PZM+94]).

Um outro aspecto importante em sistemas colaborativos, que examinaremos mais aprofundadamente em 6.5.-DIFUSÃO DE AWARENESS, é o do compartilhamento de objetos. Como diversos objetos diferentes podem embutir referências comuns, (aos mesmos objetos subjacentes), o compartilhamento das informações pode ser realizado também de uma maneira simples. O que certamente não é simples é a implementação destes mecanismos, principalmente em presença dos requisitos especiais de difusão de *awareness* que viremos a discutir.

O grande teste para qualquer modelo de objetos não é o da criação de tipos e instâncias, mas a da adaptação destes objetos de forma dinâmica, como requerido pelos sistemas de workflow. Vamos examinar a seguir o quão flexíveis são os conceitos apresentados quando adaptações se fazem necessárias.

6.3.1.3. Adaptação de objetos

Conforme já mencionado em diversas ocasiões, um aspecto fundamental em sistemas de workflow é o do suporte à adaptação dinâmica necessária para se fazer frente a contingências e evolução. Vamos ver agora que os mesmos conceitos apresentados podem ser utilizados de uma maneira natural para enfrentar estes desafios.

A resposta proposta por nós ao problema da adaptação de objetos a condições específicas consiste em não tratarmos o momento de criação de um objeto como um evento especial. A qualquer momento a composição de um objeto pode ser modificada através da sua própria janela de diálogo, de forma semelhante ao que é feito durante a sua criação. Isto deve poder acontecer mesmo posteriormente, durante o seu uso.

Suponha que um determinado candidato, o João, por exemplo, não envie o seu currículo, mas sim uma mensagem, via correio eletrônico, em que justifica o atraso no envio do currículo. Esta informação adicional, não prevista, precisa ser adicionada às informações deste candidato. Para isto, o agente que recebe a comunicação pode adicionar um novo componente ao objeto que registra os dados do João, através do arrasto de uma referência à mensagem desejada (fig. 59).

Note que a partir deste momento o objeto que armazena as informações do João passa a ter uma estrutura particular, diferente da dos objetos dos demais candidatos, aos quais também é permitido, se necessário que se diferenciem entre si de maneira não antecipada, através da inclusão e remoção de quaisquer componentes.

O fato de que os usuários normalmente sentem necessidade de aplicar operações de tipo (modificação de esquema) no momento em que estão atuando sobre instâncias é observado por Malone et al. em [MLF92], cujo sistema, Oval, também permite este tipo de funcionalidade. Neste aspecto, nos aproximamos do paradigma das planilhas, que já apresentamos como sendo mais apropriado em sistemas de workflow [SMM+94, LMY88], não exigindo uma mudança de modo de operação para que se possa efetuar modificações de esquema.

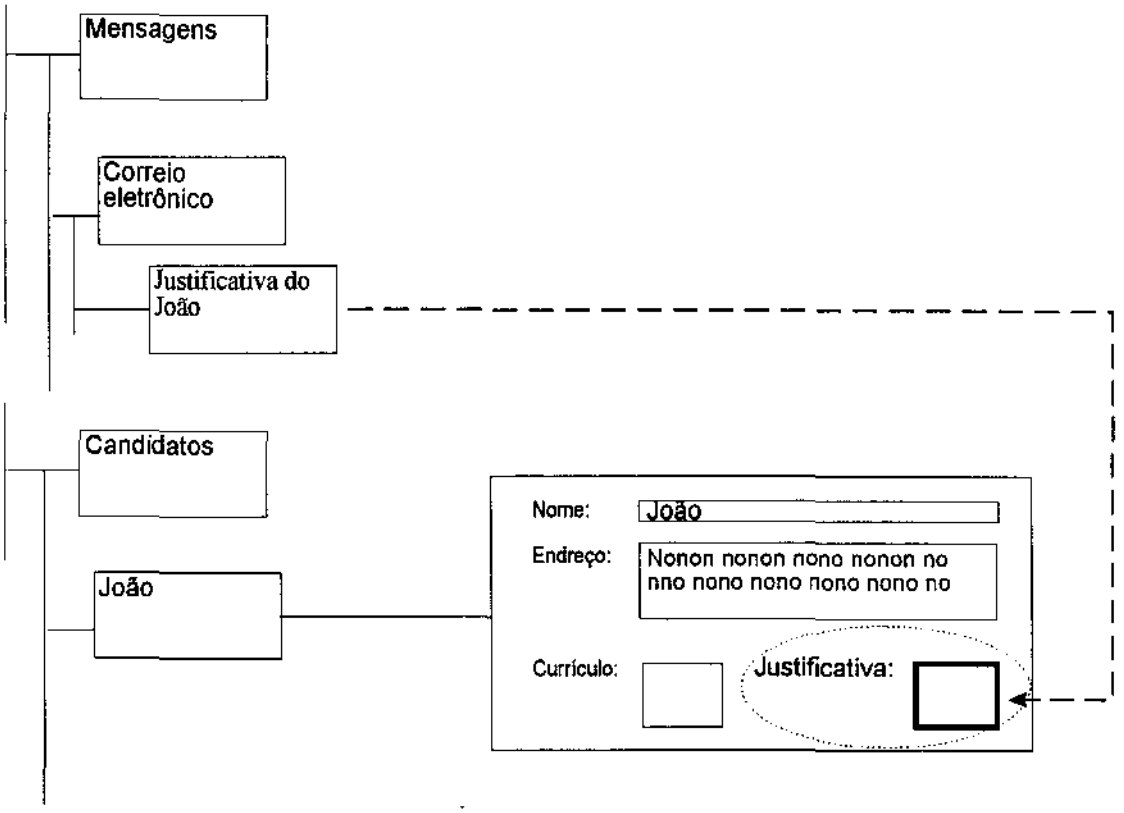


Figura 59 - Adaptação de objeto existente.

Finalmente, precisamos estabelecer o modo pelo qual são introduzidas modificações que correspondem a evoluções gerais, como no caso em que uma nova legislação torna obrigatório o registro de alguma informação adicional não prevista para todos os candidatos. Examinaremos este aspecto a seguir.

6.3.1.4. Evolução de tipos

A evolução é bastante semelhante a uma adaptação de objeto, como a que acabamos de examinar. O que diferencia a evolução da adaptação específica é apenas o grau de hierarquia do objeto modificado. Na adaptação, modificamos instâncias, e na evolução modificamos generalizações, objetos que possuem subordinados.

Quando uma determinada modificação estrutural precisa ser introduzida para toda uma classe de objetos, a conveniência da existência do objeto que corresponde à generalização se torna mais clara. Se existirem objetos subordinados, as modificações introduzidas em um objeto devem ser propagadas automaticamente para estes subordinados:

- Caso se elimine o componente "endereço" do objeto "Candidatos", por exemplo, esta eliminação precisa ser propagada para que cada objeto subordinado também elimine este componente.
- O mesmo deve ocorrer na inclusão de um novo componente. Ao incluirmos um novo componente "idade" no objeto "Candidatos", este novo componente é incluído automaticamente em todos os objetos subordinados;

- O terceiro tipo de alteração a ser propagado corresponde a modificações de atributos. Se a identificação da unidade à qual o candidato pleiteia uma vaga for modificada de "DCC" para "IC", por exemplo, todos os objetos subordinados também devem ter seus respectivos campos alterados. Lembre-se de que atributos instanciados em uma classe correspondem à faceta de objeto desta classe e são, para todos os efeitos, constantes nos objetos subordinados [PZM+94].

Note que esta propagação corresponde a alterações dinâmicas de esquema, que podem não ser simples e muito menos eficientes, se não estruturadas de maneira cuidadosa.

A difusão das modificações para os objetos de menor grau na hierarquia garante que um objeto mantenha suas características de tipo, ou seja, de modelo de uma categoria de objetos. A existência da hierarquia diretamente manipulável permite que modificações que se aplicam a toda uma categoria possam ser feitas de maneira centralizada, através da alteração de um único objeto de mais alto grau hierárquico.

O modo como é feita a difusão destas modificações estruturais deve respeitar os critérios de *awareness*, que discutiremos em 6.5.-DIFUSÃO DE AWARENESS. Em especial, as eliminações de componentes, que implicam potencialmente na perda de um grande volume de informações, não pode ser propagada de maneira cega. Deve-se poder decidir caso a caso se um objeto manterá ou não o componente, se diferenciando da super-classe.

6.3.1.5. Individualismo no uso de objetos

Discutimos em 5.8.-SUPORTE AO INDIVIDUALISMO a necessidade de se permitir que cada agente ou grupo de agentes possa moldar o sistema de acordo com suas preferências. Este requisito reconhece o fato, aliás bastante evidente, de que grupos e indivíduos diferentes executam a mesma tarefa de formas variadas. Esta observação está relacionada ao fato de que não existe um único modo correto inquestionável de se atingir objetivos, mas diversos.

Da mesma forma que não existe um plano único, também não existe um conjunto de informações que seja o ideal. Principalmente em atividades menos estruturadas, cada agente terá o seu modo particular de trabalho, que se refletirá na necessidade de manipulação de conjuntos diferentes de informação a cada momento.

Desejamos permitir que cada agente adicione componentes privativos, só acessíveis a eles, e que possa tornar ocluso alguns componentes que não lhes interessam. Como os objetos serão manipulados na maioria das vezes por um grupo de agentes no decorrer do seu processamento, queremos por um lado garantir que cada um destes agentes visualize apenas os componentes que desejar, e por outro lado, restringir o acesso de outros agentes a componentes privativos. Se determinado agente só tem interesse em examinar o currículo dos candidatos, ele deve poder tornar oclusos os demais componentes, evitando a distração. Caso deseje adicionar um comentário sigiloso, contendo uma avaliação pessoal dos dados do candidato, que não deseja compartilhar com nenhum outro agente, a privacidade deste componente também deve ser garantida pelo sistema.

A solução que propomos para esta questão de proteção e privacidade se baseia na atribuição de níveis de autorização a cada objeto e componente. Cada agente deve poder manipular apenas os componentes e ações para os quais está autorizado. Existem basicamente dois níveis diferentes de direitos:

- Modificação - neste nível, o agente pode manipular livremente o componente, modificando-o, executando qualquer ação sobre ele;
- Leitura - somente a consulta pode ser realizada. O conteúdo do objeto pode ser visto, mas nenhuma ação pode ser realizada sobre ele. Se não possuir nem ao menos o direito de leitura, tudo se passa para o agente como se o objeto simplesmente não existisse.

Propomos o uso de listas de controle de acesso (ACLs), associando diretamente as autorizações dos agentes a cada objeto, componente e ação. A autorização do objeto como um todo estabelece um default a ser utilizado por componentes que não especifiquem nada em contrário. Componentes específicos podem ter autorizações diferenciadas, normalmente mais restritas do que a do objeto como um todo.

As autorizações podem se basear em descrições de papéis, semelhantes às utilizadas para descrever os executores de atividades (veja 6.4.-SELEÇÃO DE EXECUTORES). Podemos, por exemplo, permitir o acesso livre aos objetos da classe "Funcionários", a não ser para o componente "salário", que tem visibilidade limitada, por exemplo.

A especificação pode se basear na relação de pertinência entre o agente que opera com o artefato e um conjunto qualquer de descrições de agentes, associadas aos dois direitos, o de **modificação** e o de **leitura**. Agentes com direito de modificação possuem automaticamente o de leitura. Agentes que não possuem nem ao menos direito de leitura não podem visualizar o objeto ou componente. Por exemplo, para o campo "salário", podemos especificar a proteção (utilizamos uma notação informal):

```
Leitura:       Diretores ou Presidente.
Modificação:   Gerente do departamento de pessoal.
```

A leitura pode ser realizada por diretores ou pelo presidente da empresa, enquanto que a modificação só pode ser realizada por um agente que possa desempenhar o papel de gerente do departamento de pessoal.

Admitimos também condicionais para a determinação de autorizações que se baseiam em dados dos objetos, como por exemplo o direito de visibilidade de objetos "Cheque", que pode variar de acordo com o valor dele constante:

```
Leitura:       se valor < 10000,00
                   Qualquer funcionário
               senão
                   Gerentes, Diretores ou Presidente
               fimse
```

A leitura de objetos "Cheque" cujos valores forem inferiores a 10000,00 é permitida para qualquer funcionário, enquanto que cheques com valores superiores só podem ser vistos por gerentes e superiores (de qualquer área).

Note que nossa proposta de especificação de detentores de direitos é semelhante ao mecanismo utilizado para descrever os executores potenciais na seleção de executores, ou seja, é baseado em papeis descritivos e admite condicionais.

A privacidade de informações, que é um fator importante para a própria aceitação de um sistema [Rog94], pode ser garantida através do mesmo mecanismo. Caso deseje adicionar um componente que não deseja que seja conhecido por nenhum outro agente, o usuário atribui a autorização de leitura de forma que só ele possa ter acesso ao componente.

O compartilhamento de informações confidenciais entre agentes pode ser feito de maneira semelhante. Suponha que um revisor queira comunicar de forma sigilosa aos demais uma opinião pessoal sobre um determinado candidato. Para isto, ele limita a visibilidade do componente sigiloso de forma que apenas seus colegas professores possam examinar o comentário:

```
Leitura:        Professores.
```

Já vimos os conceitos referentes à criação, adaptação, evolução e suporte ao individualismo em objetos de aplicação. Vamos discutir a seguir um último tópico referente a estes objetos, o acesso não antecipado a informações.

6.3.1.6. Acesso não antecipado a informações

A análise de situações de trabalho ([Suc87], [BN95], p.ex.) revelam que situações especiais podem gerar a necessidade de acesso a informações adicionais que não podem ser antecipadas. Na busca da causa de um determinado problema, uma grande quantidade de informações relacionadas de forma direta ou indireta com um caso precisam ser examinadas, através de combinações que não se pode prever, justamente por se referirem a exceções.

Um modelo de objetos deve prover a construção de visões que permitam aos usuários extrair e combinar informações existentes de maneira livre, preferencialmente sem a necessidade de utilização de alguma linguagem de pesquisa complexa. O trabalho de Oliveira e Medeiros no domínio de sistemas de informação geográficas (SIGs), apresentado em [OM95], parece oferecer uma base bastante adequada em relação a este requisito.

[OM95] apresenta uma arquitetura que propicia a manipulação de informações (geográficas) através da manipulação direta, via interface de usuário. Esta manipulação envolve a seleção dos elementos a serem pesquisados através de navegação pelo esquema (meta-nível), e manipulação dos predicados de pesquisa, via uma interface construída dinamicamente. O aspecto dinâmico da construção e a liberdade que oferece aos usuários nos parece compatível com os requisitos necessários em sistemas de workflow.

É preciso apenas que exista um mecanismo de reutilização que permita que as mesmas consultas construídas de forma *ad-hoc* possam ser registradas, caso correspondam a acessos executados de maneira recorrente. Novamente, estamos interessados em permitir que todas as ações executadas de forma *ad-hoc* possam ser embutidas em especificações, por opção dos agentes, caso costumem se repetir.

Em especial, é interessante que ditas pesquisas pudessem funcionar de forma semelhante às generalizações, podendo ser embutidas como componentes de outros objetos e permitindo a navegação utilizando os mesmos mecanismos padrão do sistema, através das tabelas, árvores, calendários, etc.

Terminamos a apresentação dos conceitos em relação a objetos de aplicação, e passaremos a discutir os objetos de sistema que, como veremos, são tratados através dos mesmos mecanismos. É importante salientarmos que um modelo de objetos formal necessitará definir uma série adicional de funcionalidade, relacionada, por exemplo, a construtores, tipos primitivos, integração de bases de dados e aplicativos externos, controle de versões, controle de concorrência, garantia de integridade, cuja discussão omitímos por não

representarem o foco central do presente trabalho, mas que são igualmente essenciais em sistemas de workflow.

6.3.2. CONTEXTOS DE EXECUÇÃO

A partir da base conceitual que acabamos de examinar, desejamos construir objetos de sistema que apresentem o mesmo grau de facilidade de criação, adaptação, evolução e acesso. Estes objetos serão os responsáveis pela armazenagem de especificações de processo e pelo suporte à execução das atividades. Como já mencionamos, a perspectiva de se tratar as informações de sistema de maneira uniforme em relação aos demais objetos é original e constitui, em nossa opinião, um fator fundamental para que se possa atender ao requisito de integração da modelagem e execução de forma harmoniosa.

A solução que propomos se baseia no uso de **contextos de execução**. Como discutimos em 5.6.-AMBIENTE DE EXECUÇÃO, podemos detectar a existência em diversos sistemas, na maioria das vezes tímida, de contextos, que é o termo que utilizamos para nos referir aos mecanismos de acesso aos casos de forma independente de atividades. De acordo com a classificação apresentada em 5.6, nossos contextos tem a múltipla função de *workspace*, *folder* e ambiente virtual, podendo ser utilizados de forma coletiva. O que determina qual das facetas corresponde a cada contexto é o uso que os agentes fazem dele.

A facilidade de acesso às informações, propiciada pela abordagem que propomos, constitui uma das suas conseqüências mais importantes [WB96]. Como já tivemos ocasião de discutir, as informações de cada caso são acessadas simultaneamente por diversos agentes, alguns dos quais executam atividades, outros simplesmente observam, para se manterem a par do andamento dos casos sob sua responsabilidade, interferindo apenas em situações excepcionais.

Na maioria dos sistemas, o acesso é limitado a cada momento apenas aos agentes que estão executando alguma atividade, e é feito através de uma única maneira, através da lista de afazeres (*to-do lists*) destes agentes. Isto cria dois impecilhos: 1) os observadores ficam impedidos de participar, a menos que se crie alguma atividade fantasma de que eles participem e 2) se por algum motivo todas as atividades forem encerradas ou canceladas (por força de uma exceção, p.ex.), o caso deixa de existir, já que não existirá mais caminho que leve a ele!

Como optamos por tratar todas as informações como objetos, casos podem ser pesquisados, referenciados e acessados como se fossem objetos de aplicação comuns, de forma independente da execução de qualquer atividade constante do plano: basta que se tenha o direito de acesso sobre o objeto correspondente. Sendo persistentes e referenciáveis, contextos podem ser mencionados em mensagens (do tipo: "veja como foi tratado o caso X"), sejam elas eletrônicas ou mesmo telefônicas ("acesse o contexto Y e dê uma olhada"). Voltaremos a discutir o acesso aos casos quando conhecermos melhor a estrutura dos contextos de execução, que passamos a apresentar a seguir.

6.3.2.1. Estrutura geral de um contexto

O contexto de execução, cuja estrutura básica adaptamos a partir da proposta de [HKE92], é um objeto que possui alguns componentes padronizados que reúnem as informações relevantes de um caso: a **relação de participantes**, o **plano de processamento**, a **conferência**, os **dados do contexto** (fig. 60).

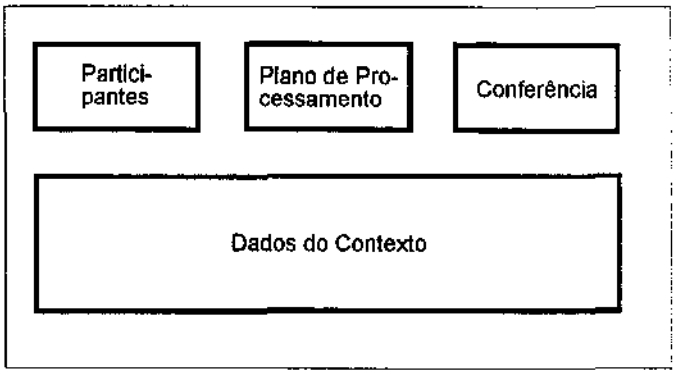


Figura 60 - Modelo genérico de Contexto de execução.

1) **Relação de participantes** - registra quais são os agentes envolvidos no contexto e o papel que exercem dentro deste contexto. Este papel pode ser livremente determinado em cada situação, podendo corresponder ao de observador, executor de alguma atividade ou qualquer outro que se julgue conveniente. Como os participantes são conhecidos, pode-se mantê-los a par do andamento dos casos através do encaminhamento de notificações (veja a Conferência, abaixo);

2) **Conferência** - de forma similar ao proposto em [HKE92], registra as mensagens relativas ao contexto. O sistema se encarrega de incluir na conferência mensagens de disparo de ações e demais eventos que surjam durante a execução de um caso. Outras notificações, geradas de forma automática ou *ad-hoc*, também são incluídas neste componente. O exame deste componente deve permitir que se tenha ideia do histórico de processamento pelo qual o caso passou até o momento. Além da parte formal, a parte informal também pode ser armazenada, representada por mensagens postadas pelos agentes. Estas informações podem ser pesquisadas em tempo de execução, de forma automática ou manual, o que permite, por exemplo, que se determine quais foram os executores de uma atividade já realizada, o momento em que foi terminada, a condição de término (normal ou cancelada) e assim por diante. O fato de que mensagens não antecipadas podem ser incluídas torna o mecanismo extensível, permitindo que seja utilizado para armazenar outras informações de estado (estruturadas ou não) que precisem ser conhecidas no futuro.

3) **Plano de processamento** - contém a especificação das etapas de processamento previstas para o caso. Este componente é o responsável pela determinação do sincronismo entre as etapas, estabelecendo as pré-condições de disparo das ações, baseado na ocorrência de eventos. Aqui encontramos a descrição de cada sub-etapa, a relação dos agentes apropriados para a execução de cada uma delas, tratamento de eventos assíncronos e assim por diante, como já examinamos em 6.2.-AÇÕES E SINCRONISMO. São admitidos planos vazios, significando que não se deseja nenhuma subdivisão explícita em etapas para o contexto. Neste caso, a determinação do modo como o trabalho será conduzido fica a cargo dos executores desta atividade.

4) **Dados do contexto ou caso** - são as informações referentes a cada contexto. Na seleção de candidatos à pós-graduação, por exemplo, os dados do caso se referem aos dados de cada candidato, como as informações pessoais, currículo e assim por diante. Este componente corresponde a uma "pasta" que contém as informações acumuladas

sobre o caso, e que tramita entre as atividades, servindo como mecanismo de comunicação implícita entre os diversos executores.

A flexibilidade dos objetos, de acordo com a base conceitual proposta, permitirá a modificação não antecipada de qualquer dos componentes, dos dados do contexto, do plano, ou qualquer outro.

A figura 61 apresenta os componentes do contexto, na notação gráfica da metodologia OMT [RBP+91], onde retângulos correspondem a classes, associações são representadas por linhas ligando as classes, agregações são indicadas por linhas de associação marcadas com losangos e especializações são indicadas por triângulos com a base voltada para as sub-classes. Utilizamos linhas pontilhadas para indicar que mais elementos podem ser adicionados opcionalmente.

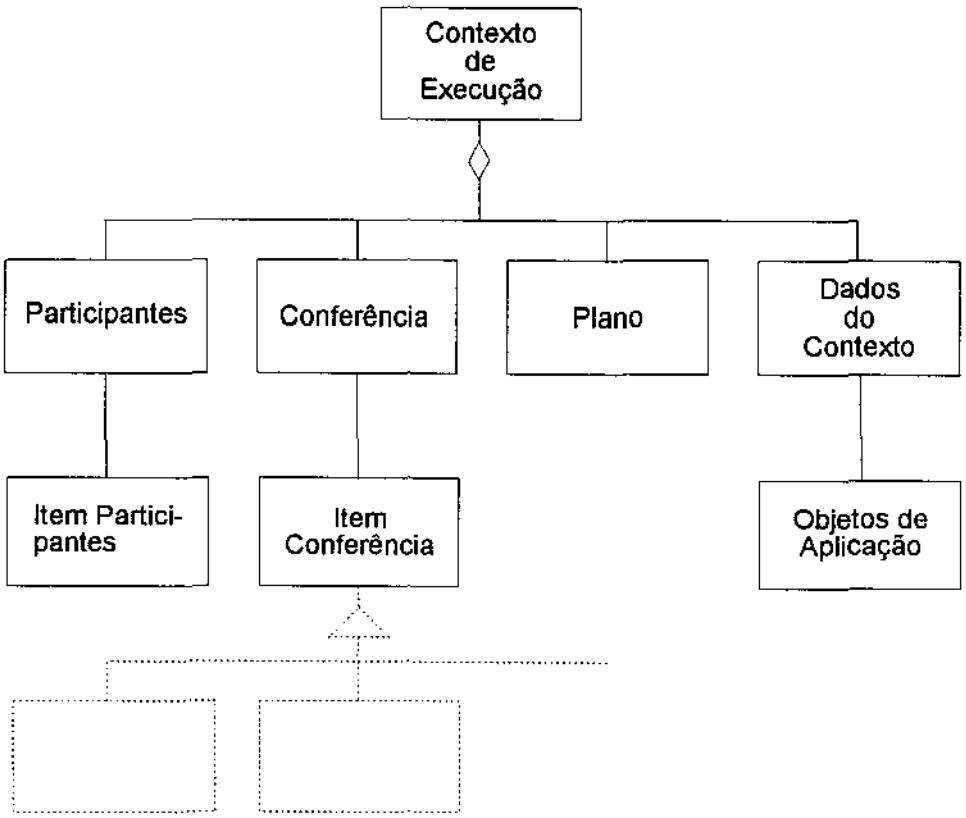


Figura 61 - Estrutura de um contexto de execução.

Da mesma forma que criamos um objeto para cada um dos candidatos da seleção de candidatos à pós-graduação a partir de um objeto "Candidatos" que funciona como um tipo, criamos instâncias de processos, que chamamos de **casos**, a partir de uma especificação expressa em um contexto de execução.

Cada candidato terá, portanto, o seu próprio contexto, que chamaremos de **contexto geral do caso**, a partir do qual as diversas etapas do processo serão disparadas, conforme o sincronismo expresso no plano de processamento. Existirão tantos contextos gerais de caso quantos candidatos (fig. 62).

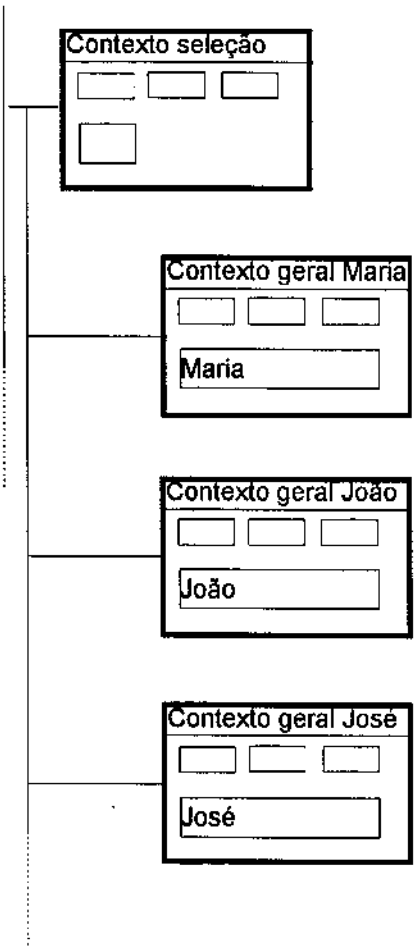


Figura 62 - Contextos gerais de caso.

A criação de uma nova instância a partir de um contexto geral de caso ocasiona diversas ações automáticas em relação aos componentes do novo contexto:

1) Criação de uma nova instância do componente "Conferência". Esta instância armazenará todas as mensagens referentes a este novo caso;

2) Criação de uma nova instância do "Plano". Esta instância será inicialmente idêntica ao plano default do processo, mas pode vir a sofrer adaptações para adequá-lo a contingências enfrentadas pelo caso, como vimos em 6.3.1.3.-ADAPTAÇÃO DE OBJETOS. Esta duplicação não é estritamente necessária, podendo-se utilizar por questões de eficiência uma estratégia de cópia tardia, postergando a criação de uma nova instância específica para quando e se houver realmente uma adaptação do caso;

3) Disparo do evento inicial do *thread* default do plano, iniciando o processamento, por exemplo, da atividade de Inscrição;

Quanto aos dados do caso, um certo cuidado precisa ser tomado. Existem processos em que as informações manipuladas são pré-existentes, já tendo sido registradas em ocasião anterior. Outros processos coletam as informações iniciais da realidade externa, registrando-as em objetos novos. Como não existe uma estratégia geral que funcione nos dois casos, optamos por não oferecer nenhum automatismo em relação ao componente "Dados

do caso". Isto pressupõe que os próprios agentes deverão se encarregar, ou de selecionar um dado de caso existente ou criar uma nova instância do mesmo, dependendo do que for apropriado. Caso desejado, a criação automática de uma nova instância de dado de caso pode ser embutida no próprio plano, através da ação apropriada (fig. 63).

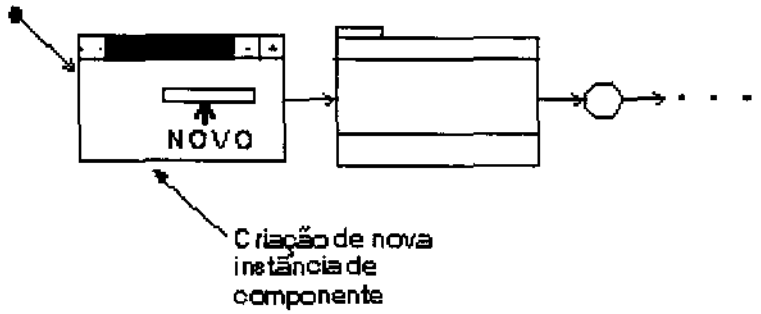


Figura 63 - Especificação de criação automática de instância de dados do caso.

Este é um exemplo da vantagem de podermos embutir nas especificações ações que são normalmente executadas apenas de forma *ad-hoc*, como a criação de instâncias de componentes.

Estes passos completam o estágio de criação de um novo caso. Passamos a descrever a lógica geral de disparo de atividades.

Cada atividade do caso corresponderá a uma instância própria de contexto. Desta forma, cada atividade pode armazenar informações próprias relacionadas a cada caso. Note que para atividades replicadas, como a avaliação, por exemplo, existirá uma instância **para cada disparo efetuado**, que normalmente serão associadas a executores diferentes (fig. 64).

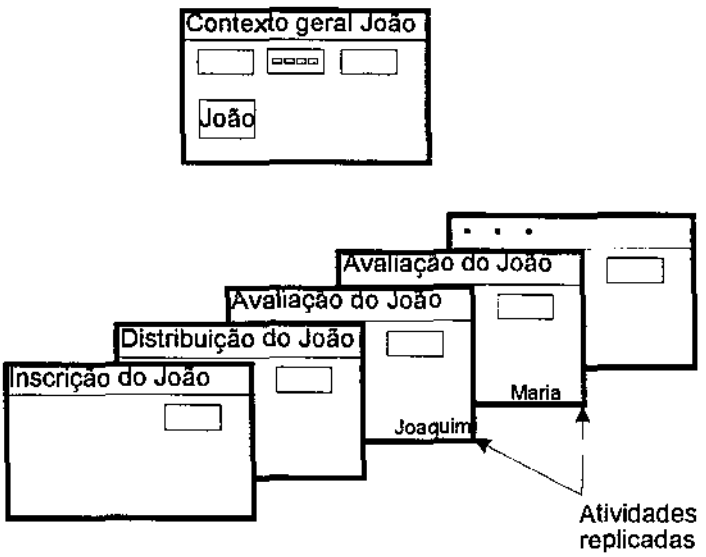


Figura 64 - Instâncias individuais de atividades.

O acesso a estes sub-contextos, criados a partir do disparo de atividades de um plano, pode ser feito através do contexto geral, permitindo fácil acesso às informações sobre o andamento do caso.

Vamos examinar em seguida como podemos definir novos contextos, adaptá-los, e evoluir as especificações, de forma semelhante ao que fizemos em relação aos objetos de aplicação (em 6.3.1.-FUNDAMENTOS PARA UM MODELO DE OBJETOS).

6.3.2.2. Definição de um contexto

A base da hierarquia de contextos é o objeto "Contextos", que estabelece a composição genérica desta classe de objetos: relação de participantes, plano de processamento, conferência e dados do caso. Este objeto será especializado para os processos específicos em objetos derivados de "Contextos", como examinaremos a seguir.

Contextos específicos são criados da forma habitual, através da ativação do método de criação de instância do objeto "Contextos", o que ocasiona a alocação de uma réplica pronta para ser especializada. Estes objetos são especializados inicialmente em relação aos dados de caso a que se referem e ao plano de processamento que contém a seqüência default de atividades, a descrição de agentes e sincronismo, especificados de forma compatível com o modo como o problema é atacado na organização (fig. 65).

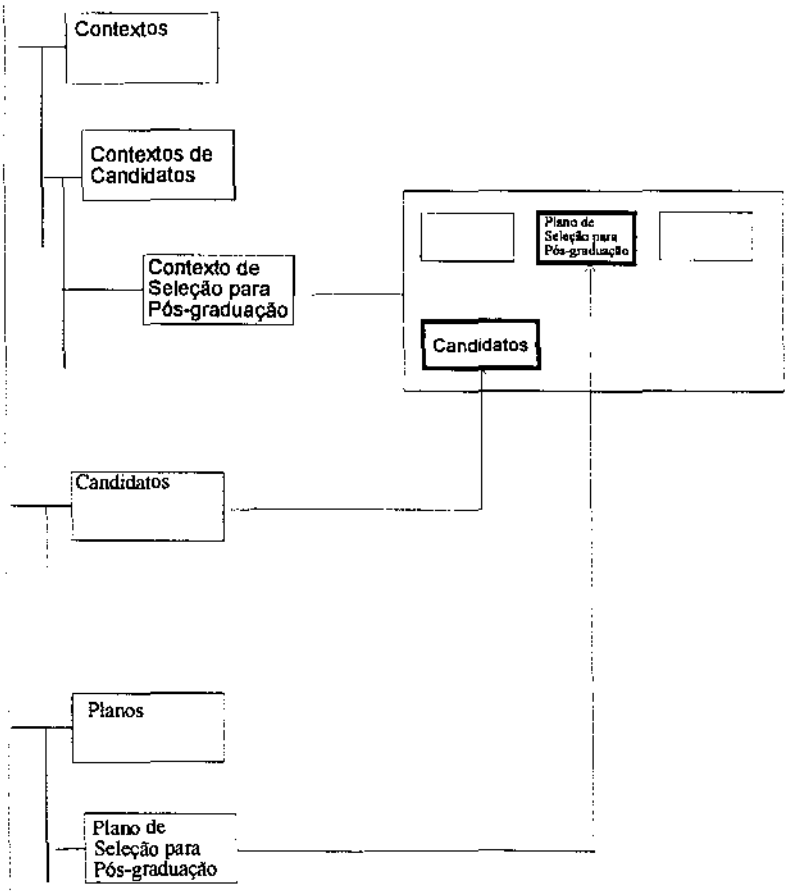


Figura 65 - Determinação do plano.

A partir deste momento, instâncias deste contexto podem ser utilizadas para o acompanhamento do caso de cada um dos candidatos. Tudo seria muito simples se não existissem as especificidades de atividades, de agentes, as contingências e a evolução, que nos obri-

gam a propiciar mecanismos adicionais de suporte. Discutiremos inicialmente como podemos registrar planos alternativos para o mesmo processo.

6.3.2.3. Planos alternativos

Nos referimos até agora ao plano de processamento de um processo, como se existisse apenas uma versão possível de processo. Como discutimos ao longo dos capítulos iniciais do presente texto, é mais comum que os processos sejam representados por uma família de especificações alternativas. Chamaremos de **plano default** aquele que representa o maior número de casos a cada momento.

Vamos representar especificações alternativas de um processo como sub-tipos deste plano default. Se existisse algum processo alternativo para a seleção de candidatos à pós-graduação, o relativo a candidatos estrangeiros, por exemplo, caso estes necessitassem de um tratamento específico por força de legislação, definiríamos este processo especial como um objeto subordinado ao de seleção (fig. 66).

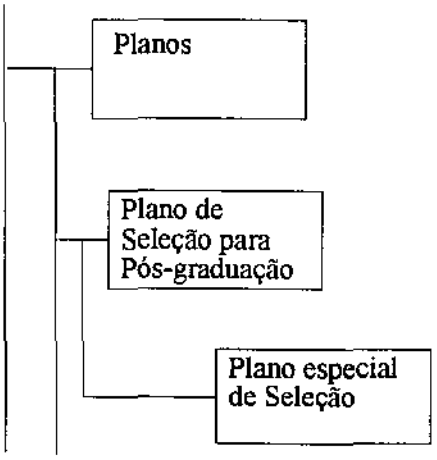


Figura 66 - Variante de plano.

O plano default não tem em princípio nenhuma primazia sobre os demais. Qualquer uma das variantes pode ser utilizada como raiz da hierarquia. É conveniente que a raiz, o plano default, corresponda à variante estatisticamente mais utilizada a cada momento, de forma a evitar que um dos sub-objetos tenha que ser selecionado muito freqüentemente. Se o volume de candidatos estrangeiros superasse o dos demais, por exemplo, este deveria passar a ser o novo plano default.

Os contextos de execução que utilizam esta família de planos normalmente incluirão uma referência ao plano raiz, a partir do qual, pelos mecanismos habituais de instanciação de atributo, qualquer uma das variantes pode ser selecionada, através de navegação a partir do componente que representa a generalização.

Já sabemos que as especificações, por mais perfeitas que sejam, estão sujeitas a contingências, decorrentes de especificidades de caso. Discutiremos a seguir este aspecto, verificando como os planos alternativos podem ser utilizados para enfrentar as exceções.

6.3.2.4. Adaptação a contingências

Alguns casos enfrentam contingências que tornam necessárias modificações específicas, não antecipadas, apenas sobre este caso particular. Dois tipos de adaptação podem ser necessárias: adaptações dos dados do caso ou do plano de processamento:

1) A adaptação dos dados do caso que apresenta uma necessidade especial se dá de forma simples, através da modificação da própria instância de candidato que armazena estas informações, a modificação do objeto com informações do João, por exemplo. A introdução e remoção de componentes de um instância correspondem aos conceitos fundamentais de manipulação de objetos propostos (veja 6.3.1.3.-ADAPTAÇÃO DE OBJETOS). A partir deste momento, esta instância passa a diferir das demais pelo acréscimo ou modificação de componentes em relação às demais.

2) Contingências que tornam necessária a modificação do plano podem ser tratadas de duas maneiras. Caso alguma das variantes de plano já armazenadas seja apropriada (veja 6.3.2.3.-PLANOS ALTERNATIVOS), basta simplesmente que o plano do contexto do caso seja trocado para esta variante. Caso exija um plano totalmente novo, uma nova variante pode ser criada, a partir de um dos planos existentes que mais se aproxime das características desejadas. Esta variante pode ser deixada disponível para futuro reuso, mesmo após o término do caso onde foi criada. Como o processo é conduzido de acordo com as informações de plano, interpretadas pelo sistema, a modificação deste componente modifica o curso de tramitação do caso. Cada caso pode, em princípio, ter seu plano específico, diferente de todos os demais e seguir, portanto, cursos específicos e diferenciados.

Veremos a seguir como é tratada a evolução de especificações.

6.3.2.5. Evolução de especificações

As modificações da realidade externa precisam eventualmente se traduzir em modificações de especificações dos contextos de execução afetados. Identificamos alguns tipos diferentes de modificações que podem se fazer necessárias:

1) Modificação dos dados do caso - o conjunto de informações relativas aos candidatos, por exemplo, pode precisar ser mudado, por força de alguma regulamentação que exija a coleta de informações adicionais, ou mudança nas informações já coletadas para todos os candidatos.

Este tipo de evolução é feito através da modificação do objeto que representa o tipo do dado do caso, o objeto "Candidato", por exemplo, conforme examinado em 6.3.1.4.-EVOLUÇÃO.

2) Modificação das etapas de processamento - quando a subdivisão representada por um plano não corresponder mais ao processo que deve ser realizado, o objeto que armazena o plano de processamento precisa sofrer modificações.

Lembre-se de que todos os contextos baseados neste plano incluem referências a este mesmo objeto, que é, portanto, compartilhado por todos estes contextos. Isto por um lado facilita a atualização, já que a modificação deste único plano afetará potencialmente todos os contextos que o utilizam. Isto pode vir a constituir um problema, caso se deseje manter para alguns dos casos uma versão mais antiga, por força de algum fator específico. Este problema é bastante semelhante ao enfrentado na evolução de objetos de aplicação, que examinamos em 6.3.1.4.-EVOLUÇÃO. Novamente, a solução é centrada no mecanismo de

difusão de awareness, que discutiremos em 6.5.-DIFUSÃO DE AWARENESS, que permitirá a escolha caso a caso da versão mais apropriada.

Caso existam planos variantes, modificações realizadas no plano default precisam ser propagadas de maneira especial para estas variantes. O mecanismo normal de propagação se oferecerá simplesmente para substituir um plano antigo pelo novo, o que não é aceitável na maioria dos casos, já que as variantes existirão justamente em decorrência de modificações sobre este plano básico. Idealmente, o próprio sistema analisa as modificações introduzidas, buscando propagá-las de forma inteligente para as variantes. Nos limitaremos no presente trabalho a supor que estas adaptações são realizadas pelos próprios agentes, quando notificados da mudança do plano default, através do mecanismo de difusão de *awareness*.

Veremos a seguir como pode ser realizado o tratamento de atividades que tenham necessidade de informações adicionais não compartilhadas com as demais atividades.

6.3.2.6. Atividades especiais

Queremos permitir que cada atividade possa pré-estabelecer, caso desejado, as informações auxiliares de que se necessita durante a sua execução, de forma que o sistema possa disponibilizá-las automaticamente no momento em que forem iniciadas.

Existem duas situações em que necessitaremos tratar atividades de maneira especial:

1) A atividade utiliza informações adicionais específicas de apoio para a realização das tarefas relativas a ela;

2) Quando a atividade é complexa e desejamos subdividí-la, por sua vez, em sub-atividades.

Nos dois casos, criaremos contextos de execução específicos, como se estas atividades correspondessem a processos independentes por si. Sobre estes novos contextos adicionamos os artefatos necessários e, caso haja divisão em sub-atividades, instanciamos o plano deste novo contexto de execução para que referencie um que contém a subdivisão desejada (fig. 67).

A simples criação de um contexto específico para uma atividade não é suficiente para garantir que ele seja acionado no momento adequado. Para isto, o sistema deve prover um mecanismo de disparo que determine qual é o contexto correto a cada momento. Examinaremos a lógica correspondente a este mecanismo em 6.3.2.8.-LÓGICA DE ATIVAÇÃO DE CONTEXTOS.

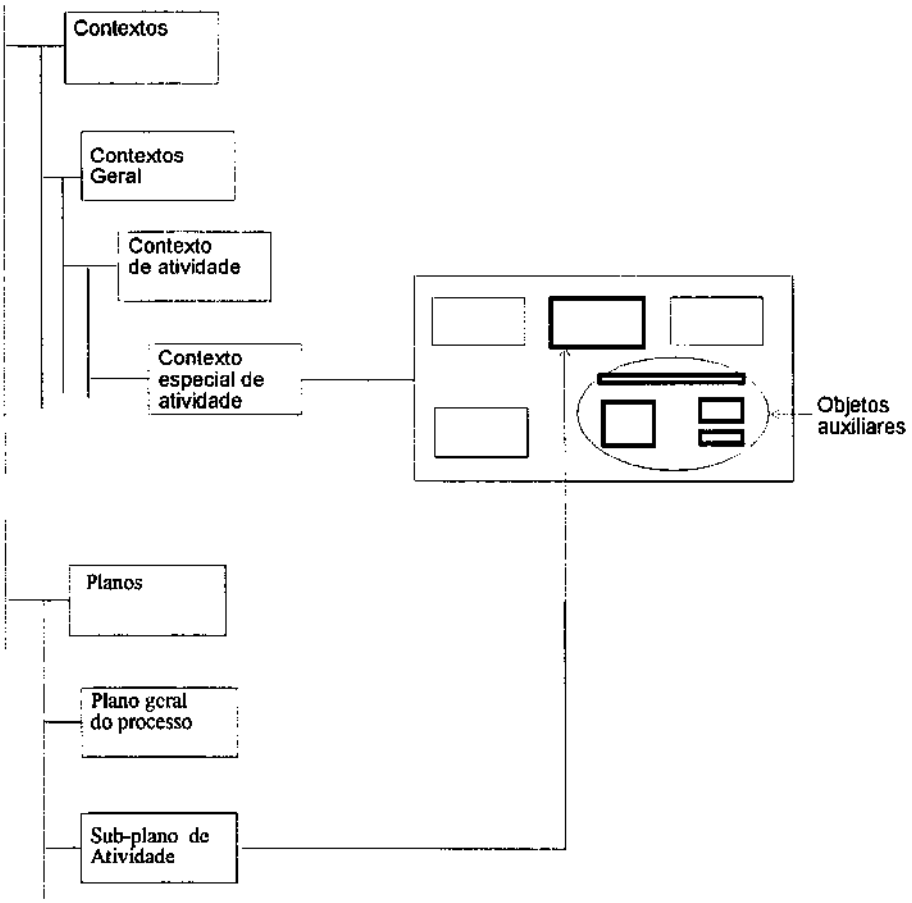


Figura 67 - Artefatos específicos de atividade.

Um outro fator que nos leva a definir contextos especiais está relacionado ao suporte ao individualismo, como veremos a seguir.

6.3.2.7. Suporte ao individualismo

Um dos requisitos estabelecidos no capítulo 5.-REQUISITOS DESEJÁVEIS, é o de suporte ao individualismo. Suporte ao individualismo significa dar liberdade aos agentes para que implementem estratégias próprias diferenciadas para atingirem os objetivos de cada atividade.

A necessidade de suporte ao individualismo vem da observação de que grupos e indivíduos diferentes executam a mesma tarefa de formas variadas. Mais uma vez, a adaptação pode se dar em dois aspectos, os dados e o plano:

1) A customização de dados pode ser relativa tanto à apresentação das informações ou quanto às informações em si. No primeiro caso, o modelo de dados deve permitir a criação, por manipulação direta, de uma apresentação particular do agente. A customização das informações está associada aos dados de apoio para a execução de atividades. Principalmente em atividades menos estruturadas, cada agente terá o seu

modo particular de trabalho, que se refletirá na necessidade de manipulação de conjuntos diferentes de informação a cada momento.

A mesma atividade de análise de candidato ao programa de pós-graduação, por exemplo, quando realizada por agentes diferentes, pode exigir um conjunto diferente de informações, de acordo com a preferência de cada revisor: um deles pode desejar consultar as informações históricas de candidatos da mesma instituição, um segundo pode fazer esta pesquisa de acordo com os autores das cartas de recomendação e assim por diante. Neste caso, a solução consiste na determinação de direitos de acesso diferenciados aos componentes, como examinamos em 6.3.2.7.-SUPORTE AO INDIVIDUALISMO.

2) A segunda fonte potencial de customizações diz respeito ao plano. Cada agente ou grupo de agentes pode estabelecer estratégias próprias de subdivisão de uma atividade complexa em etapas mais simples. Certos executores preferirão não subdividir as tarefas de modo explícito, trabalhando de forma cooperativa, enquanto que outros podem estabelecer estratégias variadas de subdivisão que se adequem à sua visão específica quanto ao trabalho.

A solução que adotaremos para propiciar o suporte ao individualismo quanto ao plano é novamente a de permitir que cada agente ou grupo de agentes crie seus próprios contextos de execução, onde estabelecem tanto as preferências em relação aos artefatos de suporte, quanto ao plano que controla a subdivisão da atividade em etapas.

Cada agente ou grupo de agentes pode criar seus próprios contextos de execução, adaptando os existentes para as suas necessidades específicas, a partir da criação de objetos subordinados aos contextos de atividade (fig. 68).

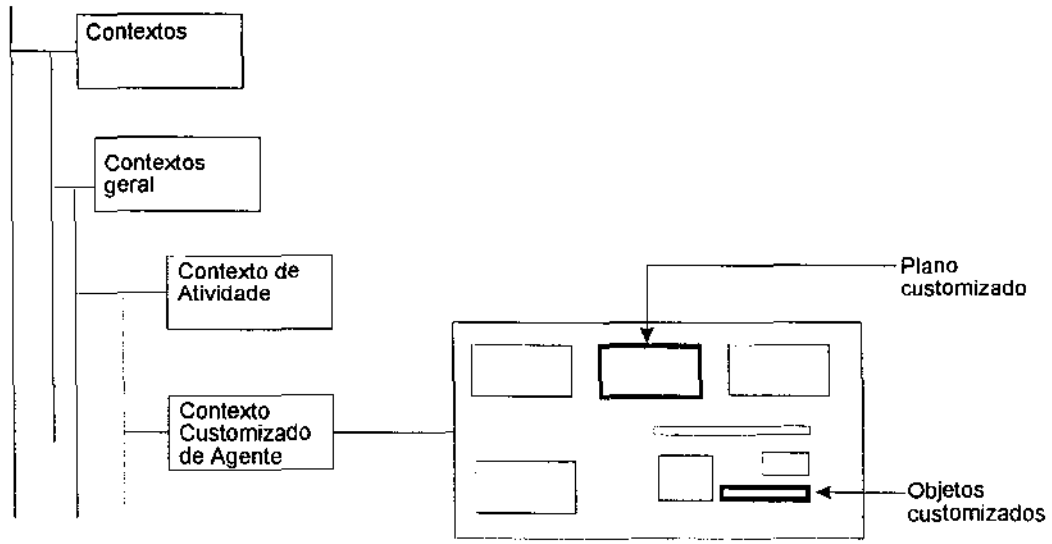


Figura 68 - Customização de agente.

Note que existirão potencialmente diversos contextos de execução variantes **para a mesma atividade**. É atribuição do sistema criar instâncias a partir do contexto apropriado. Examinaremos este mecanismo a seguir.

6.3.2.8. Lógica de ativação de contextos

Cada tipo de atividade pode ter potencialmente diversos contextos alternativos, na hipótese de existirem especializações, adaptações e customizações, conforme vimos nas seções precedentes

A cada disparo de atividades do plano, o sistema se encarrega de selecionar o contexto apropriado a partir do qual cria as instâncias para tratamento de cada caso. Apresentaremos a seguir um algoritmo abstrato que estabelece como se dá esta ativação. Nos referiremos por **super-contexto** àquele que gera o disparo da atividade e a **sub-contexto** ao da atividade disparada. O super-contexto inicial é o contexto geral do caso, a partir do qual é feito o primeiro nível de disparos.

```
A cada Disparo de Atividade em um caso
[1] Seleciona os executores
[2] Determina contexto que servirá de tipo para a atividade
[3] Cria instância do tipo determinado e inicializa Dados do caso e
    Conferência
[4] Dispara a execução
```

A seleção de executores, correspondente ao item `[1]` do algoritmo, é examinada em mais detalhes em 6.4.-SELEÇÃO DE EXECUTORES. Este passo determinará, essencialmente, quais agentes serão os responsáveis pela execução da atividade. Esta determinação inicial de agentes será utilizada na verificação da existência de contextos customizados por estes agentes, como examinaremos a seguir.

```
[2] DETERMINA CONTEXTO QUE SERVIRÁ DE TIPO
[2.1]   se existe contexto específico dos agentes para Atividade
[2.2]       tipo = contexto de agente
[2.3]   senão se existe contexto especial para Atividade
[2.4]       tipo = contexto da Atividade
[2.5]   senão
[2.6]       tipo = super contexto
[2.7]   fimse
```

A ordem de preferência é, portanto, a de agente, seguido da de atividade e por último a geral. Note que este passo garante o suporte ao individualismo, através do uso do contexto de agente customizado, e à especialização de atividades. Na ausência destes contextos específicos, é utilizado como tipo o super-contexto. O super-contexto é aquele onde foi gerado o disparo da atividade, ou seja, o contexto do nível imediatamente anterior ao da atividade disparada.

A partir deste tipo, uma nova instância será criada no item `[3]`:

```
[3] CRIA INSTÂNCIA DO TIPO DETERMINADO E INICIALIZA INFORMAÇÕES
[3.1]   Cria instância de acordo com tipo determinado em [2]
[3.2]   Dados do caso = dados do super contexto
[3.3]   Conferência = Conferência do super contexto
[3.4]   Participantes = agentes selecionados no passo [1]
```

A criação da instância é seguida de inicializações que fazem com que os "Dados do caso" passem a se referir a um caso específico, o caso do candidato João, por exemplo. De forma similar, a "Conferência" é inicializada de forma a referenciar a conferência do caso. Desta maneira, todas as mensagens relativas a um caso são armazenadas no mesmo objeto, que tem, portanto, um escopo global em relação às atividades do caso. As referências são copiadas do super-contexto para os sub-contextos, a partir do contexto geral do caso, de forma que toda a árvore de ativações, compartilha referências para os mesmos objetos de dados e de conferência.

Quanto aos participantes, passam a referenciar aqueles agentes determinados durante o passo de seleção de executores, o passo [1] do algoritmo. A partir deste momento, pode-se determinar a responsabilidade em relação à atividade do caso, através do exame da lista contida neste componente.

Resta apenas o disparo da execução propriamente dita:

```
[4] DISPARA A EXECUÇÃO
[4.1]    Notifica os participantes
[4.2]    se contexto tem plano próprio
[4.3]         Salva posição de retorno do super-plano
[4.4]         Unifica os parâmetros
[4.5]         Gera evento inicial de disparo do sub-plano
[4.6]    fimse
```

Os participantes constantes da lista de participantes são avisados de que seus serviços são solicitados, passando a ter acesso ao contexto criado. Se existir um plano específico para o contexto, este será iniciado, após o salvamento da posição de retorno do super-plano, para permitir que o processo do nível imediatamente superior possa ter continuidade quando este plano de atividade for encerrado.

A unificação dos parâmetros feita em [4.4] permite que informações sejam trocadas entre os níveis de forma explícita, como em uma chamada de procedimento em linguagens de programação. Suponha que uma atividade deva retornar uma variável *Result*, com o resultado da aprovação ou rejeição de um candidato, por exemplo. A declaração de que esta variável é manipulada na atividade permite que ela flua para os sub-níveis e retorne de uma maneira explícita.

Quando ocorrer o encerramento do sub-plano, é gerado o evento que causa a ativação do próximo elemento do super-plano, como se tivesse havido uma simples *t-action* ao final de atividade atômica. A posição de retorno, salva em [4.3] é utilizada para fazer com que as conexões apropriadas do super-plano sejam habilitadas, dando prosseguimento ao caso neste nível mais externo.

6.3.2.9. Acesso a casos e informações de sistema

Comentamos que o acesso livre às informações é essencial para se atender as variadas necessidades de informação de cada categoria de agentes, como os executores, observadores e assemelhados.

Este problema tem uma solução simples, de acordo com os conceitos que apresentamos: como todas as informações estão armazenadas em objetos, sem privilégios especiais para nenhum deles, pode-se pesquisar, criar referências e manipular qualquer objeto, mesmo os de sistema, como os contextos de execução, que dão acesso à tramitação dos casos, como se eles fossem objetos de aplicação.

Os mesmos mecanismos de pesquisa propostos para acesso a objetos de aplicação podem ser empregados para se localizar e ter acesso também aos objetos de sistema (veja 6.3.1.6.-ACESSO NÃO ANTECIPADO A INFORMAÇÕES). Os próprios contextos admitem como dados de caso qualquer tipo de objeto, o que nos permite construir alguns contextos especiais, como por exemplo:

1) Contexto de participante - cada agente precisa ter acesso aos contextos nos quais participa. Cada professor revisor, por exemplo, possui um contexto onde constam todos os casos de cuja análise ele participa. O acesso aos casos dos quais um agente participa é

oferecido pela maioria dos sistemas através de uma ferramenta especial, que representa normalmente a única via de acesso aos casos.

Note que em nosso modelo, não é estritamente necessário que o acesso dos participantes se dê através de um contexto. Uma simples pesquisa é suficiente para reunir as referências aos contextos desejados, que podem ser acessados diretamente a partir desta pesquisa. O uso de um contexto pode trazer como vantagem o fato de que o plano de processamento e a conferência propiciam mecanismos interessantes para registro de ações repetitivas e mensagens que o agente deseje utilizar.

Como admitimos múltiplos participantes em um contexto, os contextos de agente podem ser utilizados coletivamente pelos membros de uma equipe para acompanhar os contextos em que esta equipe participa.

2) Contexto de inspeção - um uso importante para os contextos é como meio para permitir o acompanhamento de um ou mais casos por parte de um supervisor ou gerente, como por exemplo, o contexto geral da seleção de candidatos à pós-graduação, que reúne todos os casos de todos os candidatos, de forma a permitir que o coordenador da subcomissão de pós-graduação possa acompanhar o andamento de todos estes casos.

Note que este tipo de contexto é bastante semelhante ao contexto de agente, com a diferença de que o critério de inclusão dos contextos observados pode variar, dependendo do interesse do observador: um gerente pode desejar examinar todos os contextos do departamento pelo qual é o responsável, ou apenas um determinado tipo de casos que exijam um acompanahmento mais próximo, por exemplo. O que vai variar é o critério de inclusão especificado na pesquisa que representa os dados do caso do contexto de inspeção.

Como já observamos em 5.6.-AMBIENTE DE EXECUÇÃO, a observação pode se transformar em ação, sempre que for detectada alguma anormalidade. Como dão acesso direto aos contextos de caso, o observador, desde que autorizado, pode interferir sempre que julgar necessário, executando as ações corretivas que julgar convenientes.

3) Contexto de planejamento - se desejado, a própria construção de especificações, que são armazenadas nos planos de processamento, podem ser conduzidas em seus próprios contextos. O planejamento do processo de seleção de pós-graduação do próximo período pode ser efetuado desta forma, de maneira colaborativa, envolvendo diversos participantes. Estes contextos de planejamento não tem nada de especial em relação a um contexto de caso, a não ser pelo fato de que os dados manipulados se referem a um plano ou contexto, e não a um candidato, por exemplo.

Este uso reflexivo é propiciado pela orientação a dados proposta, em que tanto contextos, quanto planos e demais componentes são objetos semelhantes a objetos de aplicação e podem ser manipulados de forma semelhante, sob controle de seus contextos de execução específicos.

6.4. SELEÇÃO DE EXECUTORES

Agora que conhecemos a semântica do modelo, proporcionada pelas ações e elementos de sincronismo, e o ambiente de execução proposto, vamos verificar como se processa a se-

leção de executores para as atividades disparadas pelo plano de processamento no âmbito dos contextos de execução.

A seleção de agentes é uma das tarefas fundamentais em um sistema de workflow, por determinar os agentes mais adequados à realização de cada atividade. É fácil ver o que a ausência de bons mecanismos relacionados a esta distribuição acarreta: atividades serão realizadas por agentes não capacitados ou menos capacitados. A seleção de agentes pode ser efetuada de acordo com dois critérios básicos, de forma manual ou automática.

1) Seleção manual - um agente toma para si a tarefa de estabelecer quais serão os responsáveis por tarefas subseqüentes, utilizando critérios subjetivos. A atividade de distribuição para avaliação do processo de seleção de candidatos à pós-graduação é uma atividade deste tipo (veja 6.2.5.-EXEMPLOS DE PLANOS).

Como se pode deduzir do exemplo apresentado, a seleção manual se dá pela atribuição de valores a variáveis que serão utilizadas em atividades posteriores, nas expressões de seleção de agentes, como a *Revisor*[*Iter*], utilizada na atividade de Avaliação.

2) Seleção automática - o próprio sistema se encarrega de selecionar os agentes, baseado nas informações passadas no momento do disparo da atividade. A estratégia predominante na literatura consiste na oferta da atividade a todos os possíveis candidatos, aguardando até que o primeiro deles inicie efetivamente sua execução, quando, então, a oferta feita aos demais candidatos é retirada. Esta estratégia pressupõe normalmente que apenas um único executor será escolhido para cada atividade.

Propomos o uso de um mecanismo mais geral do que o permitido pela maioria dos sistemas existentes, capaz de selecionar mais de um agente para a execução de atividades, baseado em diversas estratégias alternativas. Além de ser mais abrangente do que os existentes, este mecanismo pode ser estendido para incluir novas estratégias, se desejado. Este mecanismo consta de três etapas:

```
[1] Avaliação das descrições
[2] Ordenação por critério determinado
[3] Escolha do "n" primeiros elementos
```

No passo `[1]`, a partir de um conjunto de descrições funcionais, seleciona-se os agentes que compõem o conjunto dos candidatos à execução da atividade. As descrições que usaremos correspondem a pesquisas que selecionam um conjunto de agentes.

Na etapa correspondente ao passo `[2]`, as estratégias se diferenciam. Se a ordenação for realizada por carga de trabalho, por exemplo, temos um estratégia de balanceamento de carga (*load balancing*). Outras ordenações permitem que sejam implementadas estratégias do tipo *round-robin* (ordenando-se por um atributo de turno), ou por especialidade (ordenando-se por atributo de experiência, p.ex.).

A escolha dos "n" primeiros elementos, referente ao passo `[3]`, retira o grupo de candidatos com melhor colocação na lista ordenada pelo passo `[2]`, que serão os executores efetivos da atividade. Caso a atividade seja individual, basta escolher um único candidato, o primeiro.

Tendo tido uma visão geral da seleção, vamos discutir a seguir mais detalhadamente cada uma das etapas propostas.

6.4.1. AVALIAÇÃO DE DESCRIÇÕES

O primeiro passo do mecanismo de seleção é ativado pelo disparo de uma atividade de um plano. Como vimos (em 6.2.-AÇÕES E SINCRONISMO) a ação de disparo estabelece as descrições dos agentes candidatos a serem os executores da atividade, bem como a estratégia a ser utilizada e o número de agentes a serem empregados na execução.

Podem ser utilizadas expressões condicionais e de pesquisa na determinação destes parâmetros, permitindo que os dados influenciem na escolha dos candidatos apropriados, conforme proposta de Bussler em [Bus94]. Bussler demonstra neste artigo que existem situações em que a simples descrição de papéis não é suficiente para a correta determinação dos executores. Um exemplo utilizado em [Bus94] é o do processo de reembolso de despesas de viagem. De forma simplificada, este processo tem início pelo preenchimento de um pedido de reembolso das despesas realizadas, por parte do interessado. Este pedido é analisado, e o pagamento é creditado diretamente ao agente, caso aprovado (fig. 69).

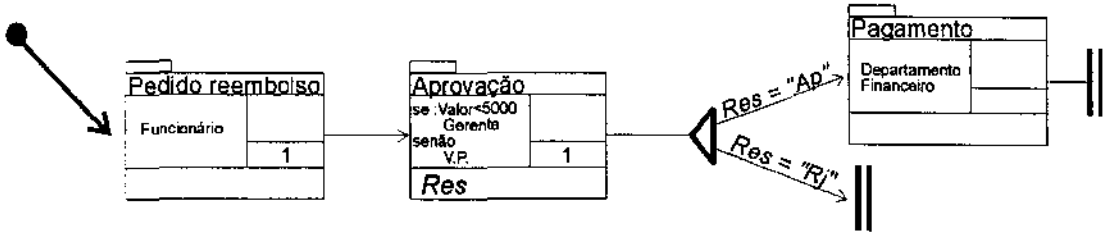


Figura 69 - Processo de reembolso de despesas.

A atividade de aprovação de reembolso deve ser executada ou por um gerente ou por um vice-presidente, dependendo do valor da despesa:

```
se Valor < 5000
    Gerente responsável pelo autor do pedido
senão
    Vice-Presidente da área do autor do pedido
fimse
```

Se o valor da despesa for inferior a cinco mil, a autorização pode ser realizada pelo gerente responsável pelo funcionário que está pedindo o reembolso. Caso seja superior a este limite de cinco mil, será preciso que o vice-presidente da área à qual o funcionário pertence faça a aprovação. Este exemplo de Bussler, apesar de atípico, na medida em que o autor do pedido é, por coincidência, também um funcionário da organização, ilustra um mecanismo que pode ser bastante útil em diversas situações semelhantes, como a aprovação de crédito no Processamento de Pedidos, examinado em 2.4.-SISTEMAS DE WORKFLOW.

O mecanismo de papéis simples utilizado na maioria dos sistemas pressupõe que qualquer gerente, ou qualquer secretária estará capacitada a realizar qualquer tarefa relativa a esta posição hierárquica, sendo indiferente a unidade em que trabalha. Isto nos leva à situação impraticável de atribuir à secretária do departamento jurídico tarefas relacionadas com o marketing, por exemplo, ou pior ainda, atribuir ao gerente de desenvolvimento a aprovação de despesas de viagens realizadas por um membro do departamento de vendas. Este tipo de atribuição fere a distribuição de trabalho da organização, que normalmente não é arbitrária, mas tem uma razão de ser.

Agentes não são máquinas, capazes de executar cegamente classes de atividades, uma vez que a execução de atividades depende de um conhecimento de contexto que só alguns agentes possuem a cada momento, como por exemplo, a necessária à aprovação de reembolso de despesa. Só o gerente do funcionário que entrou com o pedido de reembolso tem o conhecimento de contexto necessário para saber se o pedido é legítimo ou não.

A seleção dos candidatos se baseia, portanto, nos papéis interpretados por cada agente, bem como no relacionamento existente entre eles e as unidades funcionais da organização. Isto pressupõe que exista um modelo organizacional que capture estes aspectos, como discutiremos a seguir.

6.4.2. Modelo organizacional

Agentes participam de diversos comitês, equipes, departamentos e outras unidades semelhantes, desempenhando diversos papéis em relação a cada uma destas unidades (membro, chefe, etc.). A partir da modelagem destas unidades, participação e papel, será possível determinar quais são os agentes mais habilitados a executar cada atividade, pelo conhecimento de contexto e responsabilidades implícitos nestes relacionamentos.

A diversidade de modos pelos quais as organizações são estruturadas impede que uma representação fixa seja utilizada [Bus94, Pri93]. Conseqüentemente, deve ser utilizado um modelo que permita qualquer estrutura, que se adeqüe às diversas formas de organização. As próprias informações referentes a cada unidade podem variar: cada gerente pode possuir um limite de aprovação de crédito, por exemplo, ou qualquer outro atributo não antecipado.

Novamente, a orientação a dados do modelo faz com que não seja necessário o desenvolvimento de um componente especial do sistema para tratamento destas informações. Utilizaremos a própria manipulação de dados proposta como base para isto.

A estruturação que utilizaremos a seguir não é obrigatória, servindo meramente para ilustrar como se pode representar a estrutura organizacional através da construção de objetos. Cada organização necessitará definir seu próprio esquema, adequado à realidade organizacional específica. Por simplicidade, basearemos a nossa estrutura na apresentada em [Bus94].

Modelaremos a estrutura organizacional através de três elementos:

- Elementos organizacionais - descreve os componentes básicos da organização;
- Relacionamentos entre elementos - estabelece a interdependência entre os elementos organizacionais;
- Expressões organizacionais - define as expressões utilizadas nas descrições de agentes.

Vamos definir como elementos organizacionais os **grupos, papéis** e **agentes** (fig. 70):

- Os **grupos** compreendem todas as unidades da organização, como departamentos, setores, comissões, equipes e assim por diante;
- Os **papéis** estabelecem descrições que correspondem a posições hierárquicas e habilidades específicas, como gerente, secretário, engenheiro, por exemplo;

- Os **agentes** correspondem aos indivíduos do mundo real que executarão as tarefas, como o João e a Maria.

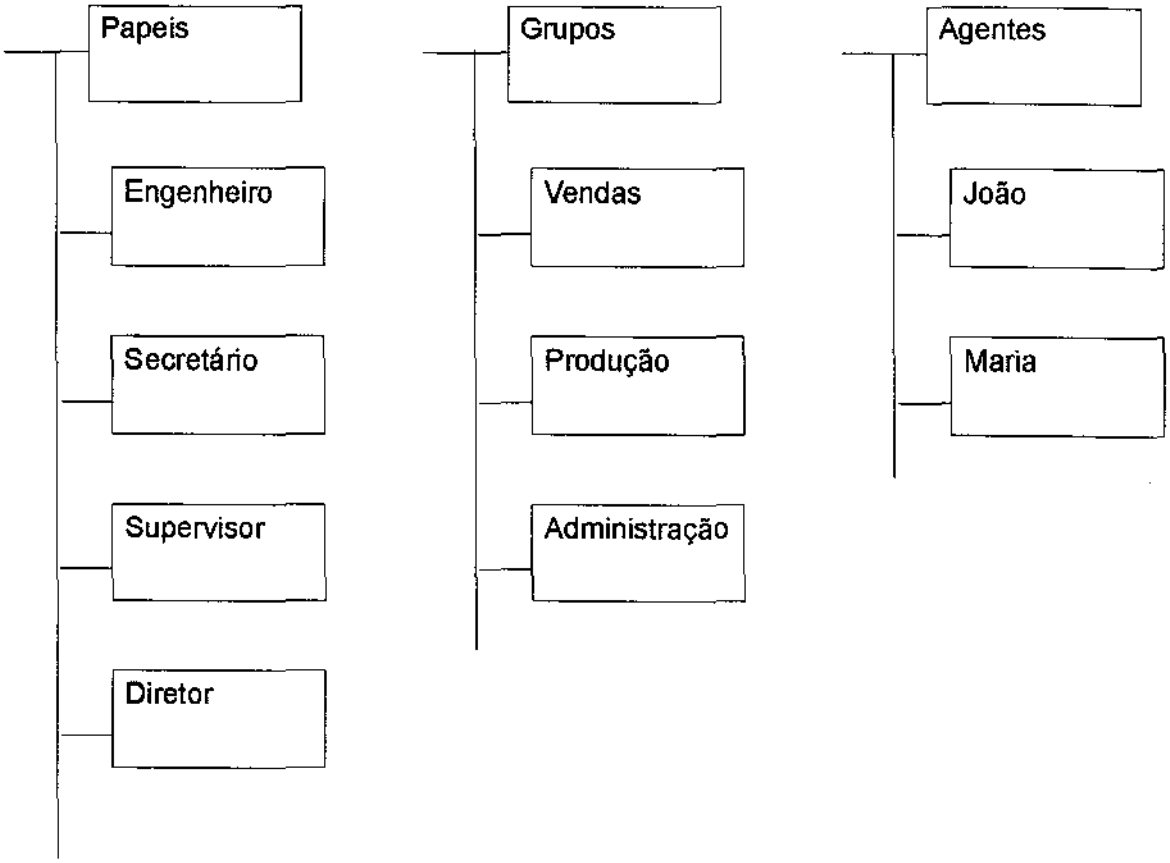


Figura 70 - Elementos organizacionais.

Os relacionamentos associam agentes a elementos organizacionais. Definimos, por exemplo, os relacionamentos **RespondePor**, **PertenceA** e **Interpreta** (fig. 71):

- **RespondePor** - associa agentes a grupos pelos quais são responsáveis;
- **PertenceA** - associa agentes aos grupos aos quais pertencem;
- **Interpreta** - associa agentes aos papéis interpretados por eles.

Finalmente, as expressões organizacionais determinam conjuntos de agentes que atendem a determinado critério, baseado em pesquisa sobre elementos e relacionamentos organizacionais. A implementação destas expressões pode ser feita com base em visões que estabeleçam população com base em pesquisas, conforme os conceitos apresentados em 6.3.1.6.-ACESSO NÃO ANTECIPADO A INFORMAÇÕES.

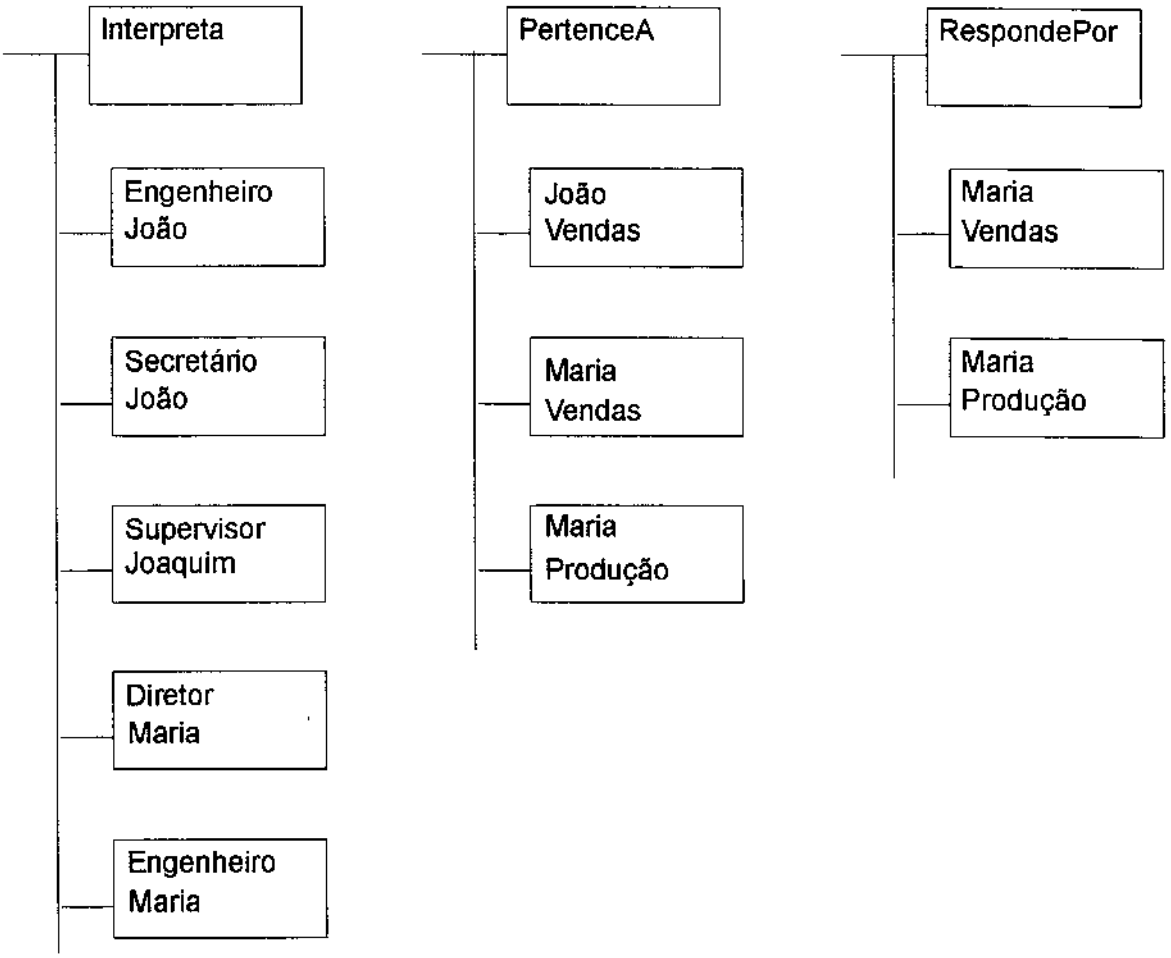


Figura 71 - Relacionamentos.

Utilizamos em nossos exemplos de expressões organizacionais uma notação ao estilo Datalog, que na prática corresponderão às consultas construídas de forma interativa, como já mencionado em 6.3.1.6.-ACESSO NÃO ANTECIPADO A INFORMAÇÕES:

```
Secretário(S) :-                Interpreta(secretário, S)
ChefeGrupo(G,A) :-              ResponsávelPor(G, A)
SubordinadosDe(A) :-            ResponsávelPor(A, G)
                                PertenceA(G, X)
                                X =/= A
```

Voltamos a ressaltar que a modelagem de uma organização pode envolver o uso de elementos diferentes e que, em especial, as expressões organizacionais são construídas com base nas necessidades surgidas em decorrência dos processo, não podendo, em princípio, ser antecipadas. Em outras palavras, as visões serão normalmente construídas para atender a uma necessidade específica de seleção de agentes, surgida quando da especificação de um processo. Como pressupomos a existência do mecanismo de visões, a determinação de novas expressões se dá pelo processo normal de definição de visões, pelos próprios usuários.

A partir das descrições, a pesquisa apropriada pode ser identificada e ativada, retornando o conjunto de candidatos correspondente. Vamos examinar em seguida como estes candidatos serão tratados no passo dois do algoritmo de seleção de candidatos.

6.4.3. ORDENAÇÃO POR CRITÉRIO DETERMINADO

Vimos como é realizado o primeiro passo, o de avaliação de descrições que correspondem a visões. Vamos agora examinar o que acontece com estes candidatos selecionados, na etapa correspondente ao passo `[2]`.

A seleção realizada na etapa que acabamos de descrever irá resultar, algumas vezes, em um conjunto de agentes com cardinalidade superior à necessária, i.é, teremos mais candidatos do que o necessário para a realização da atividade. Isto é especialmente verdade no caso de descrições genéricas, como "Funcionário", por exemplo, que se aplicam a um grande número de agentes.

Caso existam realmente mais candidatos do que a quantidade necessária, indicada no disparo da atividade, é preciso se estabelecer um critério qualquer de desempate. É neste ponto que entram em ação as estratégias alternativas. Propomos uma solução geral para este problema, baseada na ordenação dos candidatos obtidos na etapa de seleção (passo `[1]`). Esta ordenação por critérios diferentes estabelece a ordem de preferência de utilização dos candidatos, e pode ser realizada de acordo com diversos critérios:

- Carga de trabalho (*load-balancing*) - avalia-se dados de ocupação, ordenando-se os candidatos em ordem ascendente por carga. Como os dados de ocupação mantidos pelo sistema nem sempre correspondem à real ocupação dos agentes, por não considerar trabalho realizado pelos agentes fora do seu âmbito, esta estratégia pode causar distorções. Uma variante democrática permite que os próprios agentes estabeleçam a ordenação de forma indireta: uma solicitação de trabalho é enviada a todos os candidatos. Os próprios agentes aceitam em tempos diferentes as tarefas, estabelecendo a ordenação. Esta é a estratégia mais mencionada na literatura, sendo muitas vezes a única oferecida na maioria dos sistemas, com a restrição adicional de que apenas um único agente é selecionado.
- *Round-robin* - a ordenação procura fazer com que haja uma distribuição eqüitativa, colocando no início da lista os candidatos que receberam o menor número de tarefas recentemente;
- Especialidade - classifica de acordo com alguma graduação indicativa de especialidade relacionada ao caso em questão;

Apesar de que novas estratégias podem, em princípio, ser introduzidas, esta normalmente não é uma tarefa para usuários, por mais sofisticados que sejam. A principal dificuldade diz respeito ao armazenamento das informações que servirão de subsidio para o algoritmo de desempate da fase 2. Para que se possa ordenar por carga de trabalho, por exemplo, é preciso que estas informações sejam atualizadas durante a execução dos processos. No caso da carga, a cada início e término de atividade, pode-se atualizar as informações dos agentes responsáveis pela sua execução.

Uma vez ordenados, é preciso apenas retirar a quantidade desejada de agentes, como especificaremos a seguir

6.4.4. ESCOLHA DOS EXECUTORES

Vamos examinar agora o que ocorre no passo 3, em que os executores efetivos são escolhidos.

Como admitimos atividades coletivas, durante o disparo da atividade é especificada a quantidade de agentes que devem ser obtidos a partir da aplicação da estratégia. Este número pode variar de um, no caso de atividades individuais, até todos, quando todos os candidatos disponíveis serão utilizados.

Como a ordenação dos candidatos de acordo com o critério desejado já foi realizada na etapa de ordenação (passo `[2]`), basta simplesmente escolher os que obtiveram melhor colocação (os primeiros da lista classificada).

Uma otimização possível do mecanismo proposto consiste na realização simultânea dos passos `[2]` e `[3]`, podendo-se optar por busca em vez de ordenação, no caso de um número grande de candidatos e/ou número pequeno de executores efetivamente desejados. No caso de atividade individual, por exemplo, é certamente mais eficiente se buscar o elemento mais apropriado do que ordenar todos eles para então se escolher o primeiro.

6.5. DIFUSÃO DE *AWARENESS*

Examinamos nas seções passadas o aumento de expressividade de ações e de sincronismo (6.2.-AÇÕES E SINCRONISMO), o ambiente de execução baseado em dados (6.3.-AMBIENTE DE EXECUÇÃO) e a estratégia proposta de seleção de executores (6.4.-SELEÇÃO DE EXECUTORES). Vamos passar a examinar um último aspecto, o de comunicação, que optamos por tratar dentro do contexto mais abrangente de difusão de *awareness* (ciência).

O requisito de suporte ao compartilhamento dos objetos introduz necessidades ausentes em manipulações que pressupõem o isolamento entre os agentes. Objetos devem permitir concorrência de acesso e modificação, fornecendo mecanismos de difusão de *awareness* (ciência) das modificações a todos os agentes envolvidos.

O acesso concorrente, quando realizado sob a supervisão direta e simultânea de agentes não automatizados, permite que os próprios agentes se encarreguem, em muitos casos, de determinar a consistência das informações, a partir de critérios subjetivos. A proteção estrita via *locking* deve ser, portanto, substituída por mecanismos alternativos de garantia de consistência em presença de falhas de sistema e lógicas (veja por exemplo [KR95], [BDS+93], [Hsu93], [Hsu95]), aliado a mecanismos de notificação e de interface que chamem a atenção dos demais agentes para as modificações realizadas por cada um. A interface de usuário deve comportar mecanismos de exibição destas modificações com baixa interferência no trabalho independente de cada agente, conforme iremos examinar adiante.

Objetos compartilhados, aos quais adicionamos mecanismos de difusão de *awareness* podem funcionar como **artefatos comunitários**, conforme proposta de Robinson [Rob93]. Como discutimos em 3.2.-SOBRE O TRABALHO, o artefato comunitário funciona como um espaço comum através do qual o trabalho é tornado explícito, como em um quadro de chaves de hotel, ou o painel de controle utilizado no controle de tráfego aéreo. Um artefato incorpora muitos dos conceitos mencionados por autores ligados à etnografia, sendo

utilizado para suporte ao trabalho situado, propiciando transições fluidas, permitindo o uso não antecipado e servindo como base do mecanismo de difusão de *awareness*. O seu uso pressupõe um duplo nível de linguagem ("*double-level language*" [Rob93]), o implícito e o explícito, quando os agentes conversam sobre o artefato durante o trabalho de articulação.

O nível implícito corresponde às informações que podem ser deduzidas a partir da observação da seqüência de ações realizadas pelos diversos agentes sobre os artefatos, sem que haja nenhuma comunicação direta entre eles. O nível explícito envolve a troca de mensagens entre os agentes, seja via texto ou voz, e permite que estes discutam alguma questão utilizando o artefato como objeto de referência. Estes dois níveis são complementares e precisam existir para propiciar o uso adequado de um artefato.

Apesar das observações de alguns autores ([Rob93], [BR94], p.ex.), de que um volume insuspeitado de informações é passado de forma automática no nível implícito, o nível explícito será usado de forma não antecipada para a realização de trabalho de articulação, em que os agentes convergem a partir de pontos de vista diferentes e na resolução de contingências. Examinaremos cada um destes aspectos a seguir.

6.5.1. Nível implícito

Um aspecto especial introduzido pelo uso compartilhado de um artefato é o relativo à difusão das modificações realizadas por cada agente para os demais. É importante, para se manter a sinergia, que as ações de cada um se reflitam de alguma forma nas interfaces dos demais agentes que estão manipulando o artefato, concorrentemente ou não. É justamente este mecanismo distribuído que torna o artefato comunitário, permitindo que a seqüência de ações realizadas se inter-influenciem e não sejam tomadas em isolamento.

Os agentes que estão operando com um determinado artefato simultaneamente recebem as notificações, preferencialmente em tempo-real (ou o mais próximo possível), de forma que possam reagir de imediato às ações realizadas pelos demais. Não se pode garantir, porém, que todos os agentes envolvidos no uso do artefato estejam a todo o momento disponíveis para receber estas notificações. Neste último caso, assim que o agente volte a utilizar o artefato, as modificações introduzidas pelos demais agentes desde o último uso devem ser exibidas.

Chamaremos o primeiro modo, em que as notificações são entregues em tempo real ou próximo dele de **notificação síncrona**, e as exibida a *posteriori* de **notificação assíncrona**. Apesar do objetivo final ser o mesmo, a notificação síncrona e assíncrona podem apresentar algumas diferenças que discutiremos abaixo.

Seja qual for o modo utilizado, a difusão das modificações deve, dentro do possível, independer do posicionamento dos componentes nas apresentações. Como pode existir um número ilimitado de apresentações alternativas associadas a cada objeto, a notificação de modificação deve assumir um caráter polimórfico. Em outras palavras, cada componente que sofre uma mudança é notificado do fato e se encarrega de exibir por sua própria conta a indicação de modificação apropriada, independentemente da posição e apresentação específicas.

6.5.1.1. Uso síncrono

Ao mesmo tempo que se deseja manter cada agente a par das ações dos demais, não devem existir interrupções abruptas no fluxo de trabalho de cada um. Seria inaceitável, por exemplo, que a cada modificação realizada por um outro agente o sistema interrompesse o trabalho e apresentasse uma mensagem na tela dos demais. As ferramentas de uso compartilhado, como planilhas e editores colaborativos fornecem diversos mecanismos interessantes de difusão periférica, como os utilizados em Grove [EGR90], por exemplo. As modificações realizadas pelos outros agentes podem ser apresentadas envoltas em uma nuvem, que chama a atenção dos demais agentes sem interrompê-los. A informação modificada não é eliminada e cada agente pode, clicando sobre a nuvem, verificar quais foram as mudanças realizadas assim que desejar. A nuvem vai mudando progressivamente de cor, de forma que se pode perceber de relance quais são as mudanças mais recentes e quais as mais antigas [EM94].

O próprio sistema que implemente o modelo deve se responsabilizar em transmitir os eventos de cada interface às demais que fazem referência às mesmas instâncias de objetos, de forma que estas possam tornar visíveis para seus operadores o que está sendo feito pelos demais agentes.

Dois tipos de informações precisam ser difundidas: 1) modificações em elementos de dados, como campos textuais, botões de rádio e demais que impliquem em escolha de opções e 2) modificações realizadas na própria estrutura do artefato, como eliminação ou inclusão de componentes.

Nos dois casos, os mecanismos devem permitir que todos os agentes que utilizam o artefato sejam notificados da modificação, tendo ocasião de perceber o significado dela e até contestá-la, se desejarem. Isto significa, por exemplo, que um elemento eliminado do artefato por um outro agente não pode simplesmente desaparecer da tela de todos os demais. Os mesmos mecanismos de uso de nuvens, etc., precisam ser aplicados de forma a dar chance de rearticulação aos demais agentes.

Um exemplo em que a notificação é especialmente necessária é no caso de eliminação de componentes. Esta eliminação afeta um ou mais objetos (caso o objeto corresponda a uma generalização) e pode implicar na perda de um grande volume de informações importantes. O procedimento mais apropriado, neste caso, é o de simplesmente alertar os agentes que operam com o objeto, dando-lhes a chance de aceitar ou não a eliminação, caso a caso. Se não aceitarem a eliminação, o objeto passa a diferir do tipo, simplesmente como se tivesse adicionado o componente a *posteriori*.

A figura 72 apresenta um diagrama referente à difusão implícita síncrona proposta. As ações de cada agente se refletem diretamente na apresentação, além de serem difundidas para os demais via rede. De forma similar, as ações dos demais agentes são recebidas pela rede e mapeadas para a apresentação específica utilizada pelo agente.

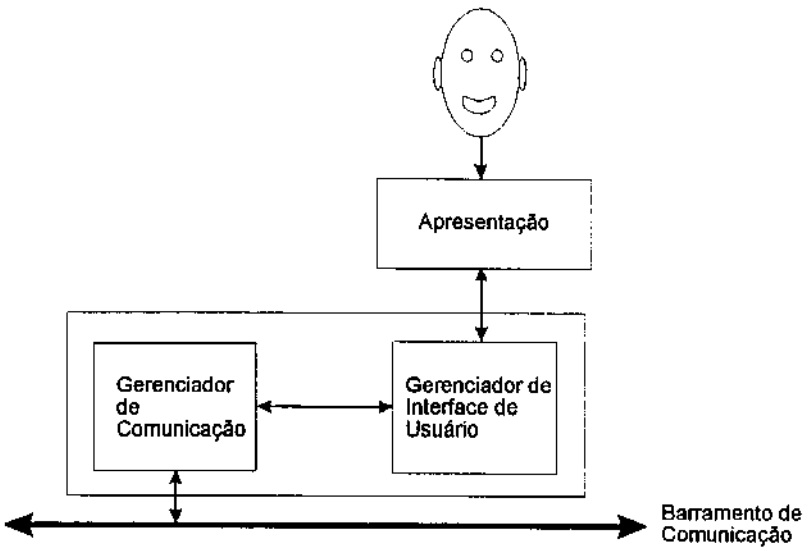


Figura 72 - Nível implícito síncrono.

6.5.1.2. Uso assíncrono

Discutimos até agora os mecanismos de *awareness* próprios para uso síncrono. Nem sempre todos os agentes participantes de uma atividade poderão estar operando com o mesmo artefato simultaneamente. Caso o agente retome o acesso a um artefato após um intervalo de tempo em que foram introduzidas modificações, estas precisam ser comunicadas a ele de alguma forma.

Mecanismos assíncronos de *awareness* são ainda raros, sendo investigados, por exemplo, no projeto wOrld [FTK95]. Uma das ideias é que o agente possa examinar a história acumulada das modificações realizadas desde seu último acesso, através de animação, onde as modificações são apresentadas de forma contínua ou "quadro-a-quadro". De qualquer maneira, o agente deve poder ter o controle sobre o modo como deseja tomar conhecimento das modificações, se de forma abrupta, ou modificação a modificação.

A figura 73 apresenta os componentes envolvidos na atualização de apresentações de forma assíncrona. Ao retomar o acesso a um objeto, o gerenciador de história é ativado, devolvendo cada uma das modificações realizadas a partir da última ativação feita por este agente. Através da interface de usuário, o agente comanda a apresentação passo-a-passo ou simultânea das modificações realizadas.

Poucos sistemas de workflow permitem a difusão de *awareness*, a não ser por mecanismos primitivos de anotações oferecidos em alguns sistemas. Este mecanismo assíncrono (utilizado em Regatta [SMM+94], p.ex.) permite que sejam associadas mensagens contendo texto, voz, ou algum arquivo, explicando a razão de alguma decisão, por exemplo.

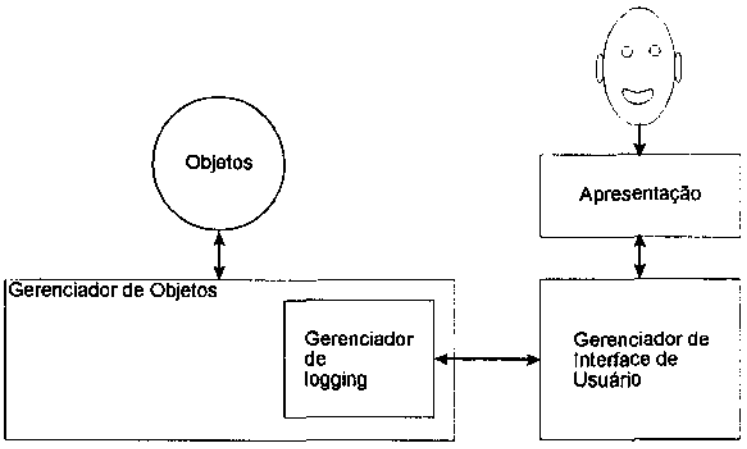


Figura 73 - Nível implícito assíncrono.

Propomos que anotações possam ser associadas a qualquer componente de um artefato. Estas anotações devem comportar pelo menos duas apresentações: 1) uma que apenas informe a existência de uma ou mais anotações associadas a determinado componente, sem interferir na interface e 2) uma que permita a visualização das anotações em si. Anotações podem ser implementadas como mais uma classe de objetos disponíveis para uso pelos agentes, de forma absolutamente semelhante a qualquer outro. O que diferencia a anotação é simplesmente a forma especial de apresentação, como o uso de *balloons*, por exemplo.

6.5.2. NÍVEL EXPLÍCITO

O nível explícito de comunicação permite que os agentes troquem mensagens para chegarem a uma posição comum em relação a algum tema relacionado ao uso de um artefato. Este trabalho pode corresponder à articulação, ao estabelecimento de convenções de uso do artefato ou da evolução do mesmo, ou pode se referir à utilização do artefato para a resolução de um problema específico.

O nível explícito está normalmente associado à comunicação síncrona, que propicia maior velocidade e *bandwidth* de comunicação, mas esta também pode ser conduzida de forma assíncrona.

6.5.2.1. Uso síncrono

Os recursos mais usuais de comunicação síncrona são os baseados em voz, vídeo e no uso de uma janela de conversação (*chat*) compartilhada [EM94, TFK95]. No nível explícito, os agentes precisam estar cientes de quais outros agentes estão atuando sobre o artefato a cada momento. Esta faceta da *awareness* envolve normalmente a apresentação da imagem de cada um dos participantes, seja ela estática (como em Grove [EGR90], p.ex.) apresentando apenas uma foto, ou dinâmica (como em wOrld [TKF95], p.ex.), através de uma imagem de vídeo com animação e voz.

O uso síncrono pressupõe que os agentes possam compartilhar uma apresentação comum, que é referenciada ou manipulada durante a sessão conjunta. Podem ser utilizadas duas maneiras de se obter este efeito: 1) através da escolha da mesma apresentação por todos os agentes e 2) através de comando que permita que a tela visualizada por outro agente

seja replicada no monitor dos demais (*broadcasting* das imagens). Caso a primeira opção seja a utilizada, é preciso que haja um mecanismo que permita a sincronização das apresentações, de forma que todos os agentes continuem a compartilhar a mesma visualização. Um cursor compartilhado pode ser utilizado para apontar, clicar e ativar o rolamento de telas simultaneamente em todas as apresentações compartilhadas.

Uma categoria específica de groupware, os *white boards*, como o nome indica, permitem que diversas anotações possam ser feitas livremente pelos agentes, como se estivessem utilizando um quadro branco comum em salas de aula. Este "quadro" é compartilhado pelos diversos agentes, que podem visualizar as anotações feitas pelos demais e contribuir com as suas próprias. Esta ferramenta está normalmente associada a seções de *brainstorming*, o que significa que as anotações são introduzidas, modificadas e eliminadas de forma dinâmica.

Propomos uma solução em que o sistema se encarregue de apresentar em cada artefato um ícone dos agentes que estão correntemente operando com ele. Estes ícones podem ser utilizados para se estabelecer comunicação síncrona com um ou mais deles. Uma ferramenta genérica de transmissão de imagem e voz pode ser utilizada para isto. Como os agentes envolvidos no uso concorrente são conhecidos (através dos seus ícones), cada usuário pode optar por entrar em contato com os demais, conforme julgue necessário, utilizando a ferramenta de comunicação. A opção pelo uso de vídeo ou voz deve ficar a cargo de cada instalação, já que pressupõe a existência de infra-estrutura específica (se não houver condição para transmissão de vídeo, pode-se optar por transmissão de imagem estática, ou apenas de voz).

As funções de *white board* e janela de conversação são automaticamente decorrentes do suporte à comunicação implícita inerente aos artefatos. Um componente com campo textual multi-linha pode ser utilizado como janela de *chat*, já que as modificações de cada um se refletem nas apresentações dos demais. O compartilhamento de desenhos e anotações à mão livre podem ser realizados através de um componente que permita edição de gráficos.

A sincronização de apresentações pode ser realizada através de negociação entre os próprios agentes. Através de troca de comunicação, podem decidir sobre a apresentação mais apropriada, para a qual todos trocam. Um objeto especial, o **cursor compartilhado**, pode ser manipulado por qualquer um destes agentes utilizando os seus cursores locais. O agente que está com o controle do cursor compartilhado pode arrastá-lo com o seu *mouse*, ocasionando a movimentação equivalente nas apresentações dos demais agentes. Ações realizadas sobre este cursor (*clicks* e *clicks* duplos, por exemplo), são também transmitidos para todos as apresentações. O controle deste cursor compartilhado é realizado por convenção social estabelecida livremente entre os agentes. Ellis e Maltzahn observam que os protocolos sociais desenvolvidos entre os agentes durante o uso de artefatos compartilhados são altamente sofisticados [EM94 p. 20], sendo portanto superiores a qualquer esquema que possa ser oferecido por um sistema de forma automática.

6.5.2.2. Uso assíncrono

O nível explícito assíncrono também é possível e é representado por mensagens de texto, voz e vídeo armazenadas em caixas postais ou secretárias eletrônicas [TKF95]. Dentro do possível, devem ser empregadas as ferramentas habituais de manipulação de mensagens

utilizadas pelos usuários, i.é, não se deve obrigar os usuários a aprenderem o manuseio de uma segunda ferramenta específica do sistema [KHK+91] para poderem receber suas mensagens.

6.6. CONCLUSÕES

Apresentamos ações e elementos de sincronismo que ampliam o poder expressivo do modelo em relação a sistemas existentes, tratando eventos assíncronos, atividades batch e replicação de atividades. Sustentamos que um dos pressupostos básicos para o bom funcionamento de um sistema de workflow consiste na possibilidade de execução de ações tanto de forma *ad-hoc* quanto sob domínio de uma especificação, a critério dos agentes.

Propusemos um ambiente de execução, fortemente orientado a objetos, onde tanto objetos de aplicação quanto objetos de sistema são tratados de maneira uniforme, propiciando a facilidade de acesso a qualquer tipo de objeto, sua adaptação, evolução e suporte ao individualismo. Uma base conceitual de gerenciamento de objetos capaz de dar suporte a estes requisitos foi apresentada.

Vimos como a seleção de executores pode ser realizada de uma forma mais flexível do que a possível na maioria dos sistemas existentes, através da aplicação de estratégias de seleção diferenciadas e principalmente a possibilidade se escolha de mais de um executor para atividades coletivas.

Finalmente, discutimos os conceitos relacionados à difusão de *awareness*, raramente associada a sistemas de workflow, e mostramos como pode ser utilizada como substrato de comunicação que se encarrega de manter cada agente a par das ações dos demais, propiciando a sinergia que é essencial em trabalhos colaborativos, e que é especialmente ausente em sistemas de workflow existentes.

7. CONCLUSÕES E TRABALHOS FUTUROS

Apresentamos uma cobertura abrangente do problema relacionado à construção de sistemas de workflow, relacionando a causa do fracasso dos sistemas pioneiros e dificuldades dos sistemas atuais a fatores estruturais e a visões de mundo parciais, que identificamos com as linhas funcionalista e interpretivista.

Mostramos que os sistemas da área de suporte ao trabalho colaborativo se dividem em duas grandes correntes, uma das quais enfatiza o aspecto guardião, associado aos dados, e comunicador, representado por ferramentas como editores e planilhas colaborativas e tele-conferência, por exemplo, e a outra corrente, que enfatiza o aspecto sincronizador em detrimento dos demais. Sistemas de workflow se enquadram nesta segunda corrente.

Sustentamos que a divisão explícita e clara entre as fases de modelagem e de execução, especialmente se realizadas por grupos diferentes de pessoas, introduz dificuldades, já que o planejamento e a execução acontecem de forma entremeada e por vezes simultânea, ocasionando a necessidade de uma mudança de paradigma.

Apresentamos um modelo conceitual que busca equilibrar os aspectos guardião, comunicador e sincronizador, tornando viável a execução e modelagem concomitante, de uma forma transparente, através de uma forte orientação a dados. Foram contemplados o aumento de expressividade, o ambiente de execução, a seleção flexível de múltiplos executores e a difusão de *awareness*, responsável pela manutenção da sinergia.

A principal e mais óbvia extensão é a de se produzir um modelo detalhado e uma arquitetura a partir dos conceitos apresentados, construindo-se um protótipo que valide na prática estes conceitos. Esta tarefa não é trivial e envolve um grande esforço de pesquisa principalmente relacionado ao gerenciamento de objetos conforme os requisitos apresentados.

Em especial, não discutimos aspectos sobre tipos primitivos, construtores, versões, construção de interface, mecanismos de garantia de integridade, a integração de aplicativos e bases de dados existentes e execução em ambientes heterogêneos e distribuídos, todos fundamentais em um modelo completo de gerenciamento de objetos que vise ser utilizado na prática.

Também o modelo de execução do mecanismo de sincronismo do modelo precisa ser detalhado. Uma proposta interessante neste sentido é apresentada por Casati et al. em [CCP+95b], onde regras ativas são propostas como mecanismo de implementação. A orientação a eventos de nossa própria proposta torna este tipo de mecanismo bastante atraente.

Um detalhe para o qual não apresentamos uma solução satisfatória é o relativo à difusão de modificações de planos de processamento de mais alto nível para as suas variantes. Uma proposta que parece promissora neste aspecto é apresentada por Bogia e Kaplan em [BK95]. Estes autores resolvem este problema através da construção das variantes através da composição de *templates* que especificam modificações sobre uma malha pré-existente. Cada *template* pode adicionar, remover e modificar componentes existentes, adaptando-os à necessidade específica do plano especial. Mesmo assim, algumas modificações nas malhas básicas requererão a intervenção humana, caso as modificações tenham sido muito extensas.

ANEXO 1 - EXEMPLOS DE PLANOS DE PROCESSAMENTO

Apresentaremos a seguir os planos de processamento referentes a alguns dos exemplos abordados em 2.6.-EXEMPLOS DE SITUAÇÕES DE TRABALHO.

1.1. AQUISIÇÃO DE INSUMOS

Este processo descreve os passos de aquisição de material realizados por uma organização. Em linhas gerais, uma solicitação interna de compra é gerada, requisitando alguma espécie de material ou insumo. Seleciona-se em seguida, a partir de um catálogo de fornecedores do produto desejado, um ou mais fornecedores. Dentre estes é selecionado o que oferecer as melhores condições comerciais (preço e entrega), sendo emitida uma ordem de compra, que é enviada ao fornecedor. Quando a mercadoria e a fatura referente a esta aquisição, é feita a programação do pagamento ao fornecedor (Fig. A-1).

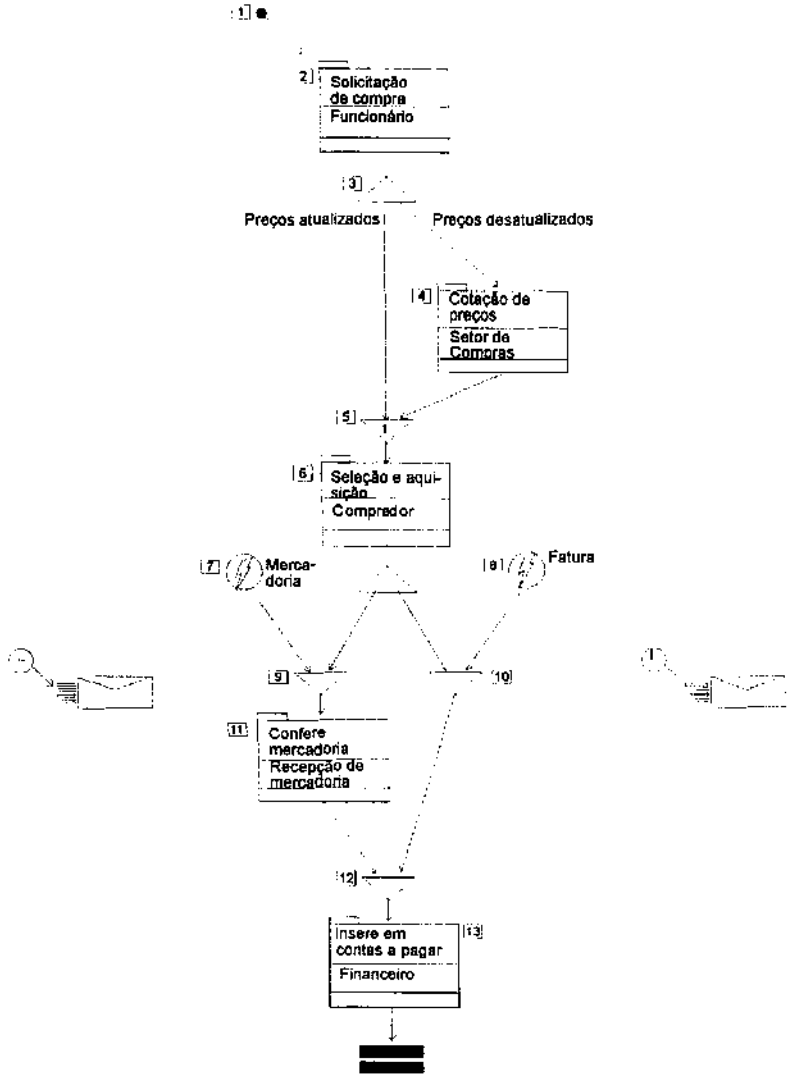


Figura A-1 - Aquisição de insumos.

Após a solicitação de compra, uma atividade opcional de cotação de preços (4) é realizada apenas se as cotações existentes estiverem desatualizadas (com data de validade vencida). O *join* com *quorum* 1 funciona como um *or-join*, habilitando a saída assim que um único sinal é recebido.

A partir destas cotações, é feita a seleção e aquisição (em 6). Dois tipos de eventos passam a ser esperados, a chegada da mercadoria e da fatura, que podem acontecer em momentos diferentes.

Assim que um dos eventos esperados chega (7 ou 8), acontece conferência da mercadoria ou a inclusão da fatura nas contas a pagar. O caso só termina quando ambas as atividades tiverem sido completadas, o que é garantido pelo *join* total 13, que só habilita o término quando ambas as atividades tiverem sido terminadas.

O atraso na chegada da mercadoria ou da fatura pode gerar as notificações nos eventos assíncronos associados aos *joins* 9 e 10, onde são esperadas.

1.2. Revisão de artigos

Na revisão de artigos, como acontece na preparação de uma conferência ou de uma revista científica, diversos artigos são submetidos, sendo analisados independentemente por um determinado número de revisores. Baseado nos laudos de avaliação e possivelmente em outros critérios subjetivos (volume mínimo de artigos necessários, p.ex.), o editor determina se o artigo será ou não aceito para publicação (fig. A-2).

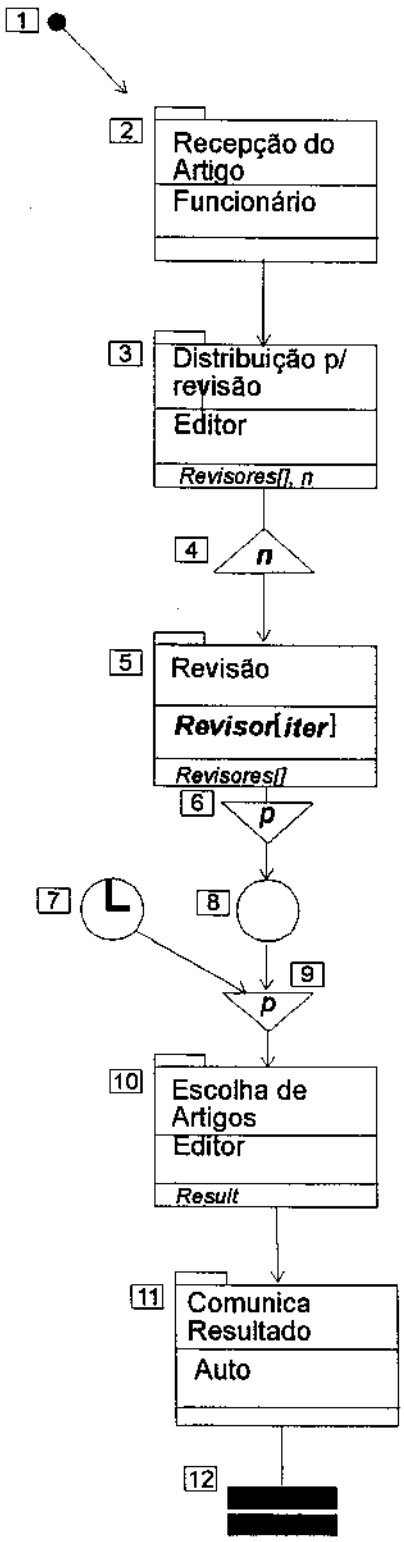


Figura A-2 - Revisão de artigos.

Como podemos perceber, este processo é bastante semelhante ao de seleção de candidatos à pós-graduação, que examinamos em ocasiões anteriores. Após a recepção do artigo (em 2), o editor seleciona manualmente um conjunto de revisores, que executam em paralelo

uma avaliação do artigo. Assim que um determinado número mínimo **p** de revisores tiver produzido seu laudo, a etapa de revisão é encerrada.

Todos os artigos são sincronizados pelo batch 8, estando à disposição do editor a partir deste momento. Quando chegar a data de escolha dos artigos, o temporizador 7 habilita a execução da próxima etapa, em que o editor aplica critérios subjetivos para selecionar um certo número deles. O resultado desta escolha é carregado pela variável *Result*, cujo valor orientará a geração de uma mensagem de aprovação ou rejeição.

1.5. CRIAÇÃO DE UM NOVO PRODUTO

Este processo, devido a Swenson et al. [SMM+94], é iniciado por um pedido genérico proveniente da presidência da organização, endereçado à divisão de desenvolvimento de produtos da organização.

O atrativo fundamental deste processo consiste justamente na sua generalidade radical, o que torna a construção de uma especificação a *priori* se não impossível, bastante difícil. A variedade de possibilidades de condução deste processo, faz com que as etapas, especialmente no início do processo, tenham que ser desenvolvidas à medida em que são necessárias ("*on-the-fly*"). Conforme o caso progride, será eventualmente possível se estabelecer etapas mais padronizadas, para execução por pessoal de linha de frente, os projetistas em si, pessoal de marketing e assim por diante.

A forma que propomos para a condução deste processo é a construção de um contexto de execução cujo plano de processamento e dados do caso são inicialmente vazios. O contexto é utilizado como ambiente virtual, facilitando a comunicação dos participantes, que vão adicionando informações (diagramas, atas de discussão, p.ex.) ao contexto à medida que seu entendimento do problema progride.

Quando e se um sub-divisão em etapas puder ser definida, ela pode ser registrada no plano de processamento do contexto, que passa a seguir estas etapas. O plano não precisa ser necessariamente completo, indicando apenas as atividades de mais alto nível. Assim que cada uma das atividades for atacada, novas sub-divisões podem ser determinadas e incorporadas a planos de processamento dos contextos de cada uma das sub-atividades.

ANEXO 2 - FREQÜÊNCIA DE OCORRÊNCIA DE EXCEÇÕES

Uma pesquisa apresentada em [SMS94] revela algumas características interessantes sobre as exceções. Esta pesquisa, realizada em 1993, apresenta resultados relativos a 93 instituições públicas e privadas da Finlândia, variando de pequenas (menos de 100 empregados) a grandes empresas (mais de 500 empregados).

Apesar do seu escopo localizado, julgamos que os resultados podem contribuir para o entendimento da distribuição de contingências de acordo com diversas categorias.

Origem da exceções

Relacionados à Organização	36.4%
Geradas no Mercado (*market born*)	3.4%
Problemas técnicos	53.4%
Outros	6.8%

Figura A-3 - Origem das exceções.

A tabela da figura A-3 apresenta como fonte principal de exceções as relacionadas a problemas técnicos, principalmente falhas de hardware e software. A segunda maior fonte são os problemas relacionados ao próprio funcionamento da organização, principalmente enganos cometidos pelos próprios funcionários. Juntas, representam 89.8% do total de exceções.

Uma conclusão da análise destes dados é a de que o impacto de modificações geradas fora do âmbito da organização não é tão grande. A segunda conclusão é a de que a alta incidência de problemas técnicos impõe um limite à possibilidade de redução de exceções, já que pouco se pode fazer no âmbito de um sistema automatizado para prevenir ou remediar este tipo de contingência.

Freqüência

Infreqüente	84.4%
Temporária	2.3%
Periódica	8.9%
Permanente	4.4%

Figura A-4 - Freqüência das exceções.

O número esmagador de exceções infrequentes demonstra segundo [SMS94], que adaptações são feitas de forma a evitar novas ocorrências da mesma exceção. Este dados é coerente com o dado apresentado na figura A-5 - impacto da exceção, onde se pode observar que em apenas 27.7% dos casos nenhuma modificação de especificação é feita. Existe, portanto, uma reação da organização visando evoluir as suas práticas.

Impacto da exceção

Nenhuma regra é modificada	27.7
Regras especiais são criadas	56.7
Regras gerais são modificadas	15.5

Figura A-5 - Impacto das exceções.

A pesquisa apresentada em [SMS94] examina o impacto de exceções de forma geral, e não ligadas diretamente ao uso de sistemas de workflow (ou a qualquer outro sistema automatizado). Como estas regras são a fonte a partir da qual as especificações de workflow serão construídas, é razoável se supor que os dados apresentados possam representar o impacto em relação a alterações de especificações nestes sistemas.

A criação de regras especiais corresponde à criação de uma nova variante de especificação feita sob medida para incluir o tratamento de alguma exceção.

A modificação de regras gerais corresponde à alteração da especificação da linha principal (*main-line*), significando que daí para a frente todos os casos seguirão estas novas regras, na pratica alterando daqui para frente a noção do que é normal e do que não é, correspondendo a uma evolução.

Consideramos que a diferenciação da linha principal em relação às outras variantes de especificação existentes não é significativa. Preferimos considerar que diversas especificações para um processo podem coexistir, cada qual apresentando um percentual de utilização diferente, que varia ao longo do tempo. A linha principal corresponderia neste caso à especificação mais utilizada em determinado momento.

Segundo este critério, podemos dizer que o impacto das exceções é incorporado a especificações em 72,3% dos casos, um alto grau de aproveitamento de experiências passadas.

Influência na organização

Nível de funcionário	3.3%
Nível de grupo	54.4%
Nível de escritório (*office*)	1.1%
Nível de organização	41.1%

Figura A-6 - Influência na organização.

Os dados relativos a grupo e organização demonstram que contingências geralmente envolverão mais de um agente na sua solução. [SMS94] questiona o baixo volume de contingências a nível de funcionário e supõe que isto se deve ao fato de que apenas exceções consideradas mais graves foram reportadas nos questionários. No caso de exceções mais graves, é natural que o funcionário que detecta o problema requeira auxílio. De qualquer forma, é interessante se perceber que o tratamento de exceções significativas envolva uma quantidade maior de participantes, com uma grande abrangência entre unidades organizacionais, como revela o alto índice associado à organização como um todo.

REFERÊNCIAS BIBLIOGRÁFICAS

[AGL+93] C.S. Amarvadi, J.F. George, O.R. Liu Sheng, and J.F. Nunamaker. The adequacy of office models. Working Paper submitted to ACM Computing Surveys, May 1993.

[ALP+94] Martin Ader, Gang Lu, Patrick Pons, Josep Monguio, Lose Lopez, Giorgio De Michelis, M. Antonietta Grasso, and George Vlondakis. Woo{RKS}, an object oriented workflow system for offices. Ithaca technical report, Bull S.A., T.A.O. S.A., Universitá di Milano, and Communication and Management Systems Unit, 1994. ftp://cui.unige.ch/OO-articles/ITHACA/WooRKS.

[AS94] K. R. Abbot and S. K. Sarin. Experiences with workflow management: Issues for the next generation. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*. ACM, 1994.

[Bar96] V.A.C Barthelmess. Ciência, Mito e Reflexão. No prelo, 1996.

[BC88] J. Bowers and J. Churcher. Local and global structuring of computer mediated communication: developing linguistig perspectives on cscw in cosmos. In *Conference on Computer-Supported Cooperative Work*, pages 125--139. ACM, September 1988.

[BDS+93] Y. Breibart, A. Deacon, H.J. Scheck, A. Sheth, and G. Weikum. Merging application-centric and data-centric approaches to support transaction-oriented multi-system workflows. Technical report, Dept. of Computer Science, ETH Zurich, 1993. ftp://ftp.inf.ethz.ch/ pub/publications/papers/is/ dbs/bdssw93-sr93.ps.Z.

[BF95] C. Barros Filho. *Ética na Comunicação*. Editora Moderna, São Paulo, 1995.

[BK95] D.P. Bogia and S.M. Kaplan. Flexibility and control for dynamic workflows in the worlds environment. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 148--159, Milpitas, California, August 1995. ACM.

[BN95] R. Blumenthal and G.J. Nutt. Supporting unstructured workflow activities in the bramble icn system. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 130--137, Milpitas, California, August 1995. ACM.

[BR94] S.G. Blair and T. Rodden. The opportunities and challenges of cscw. *Journal of the Brazilian Computer Society*, 1(1), July 1994.

[Bul92] Bull Corporation, Paris, France. *FlowPath Functional Specification*, September 1992.

[Bus94] C.J. Bussler. Policy resolution in worklow management systems. *Digital Technical Journal*, 6(4):23--49, Fall 1994.

[BW95a] P. Barthelmess and J. Wainer. Workflow systems: a few definitions and a few suggestions,. In *Proceedings of the ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 138--147, San Jose, CA, 1995.

[BW95b] P. Barthelmess and J. Wainer. Workflow modeling. In *CYTED-RITOS International Workshop on Groupware*, Lisbon, Portugal, September 1995.

[CCP+95a] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Conceptual modeling of workflows. Technical report, Dipartimento di Elettronica e Informazione - Politecnico di Milano, 1995. ftp://www.elet.polimi.it/pub/papers/Giuseppe.Pozzi/OOERfinal.ps.

[CCP+95b] F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Deriving Active Rules for Workflow Enactment. Technical report, Dipartimento di Elettronica e Informazione - Politecnico di Milano, 1995. http://www.elet.polimi.it/~pernici/activewf.ps.

[DW91] J.L.G. Dietz and G.A.M. Widdershoven. Speech acts or communicative action? In L.Bannon, M. Robinson, and K. Schmidt, editors, *Proceedings of the Second European Conference on Computer-Supported Cooperative Work*, pages 235--248, Amsterdam, The Netherlands, September 1991. Kluwer Academic Publishers, Dordrecht.

[EB82] C.A. Ellis and M. Bernal. Officetalk-d: An experimental office information system. In *Proceedings ACM SIGOA Conference on Office Information Systems*, pages 131--140. ACM, June 1982.

[EGR90] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Design and use of a group editor. In G. Cockton, editor, *Engineering for Human Computer Interaction*. North-Holland, Amsterdam, 1990.

[EGR91] C. A. Ellis, S. J. Gibbs, and G. L. Rein. Groupware: Some issues and experiences. *Communications of the ACM*, 34(1), 1991.

[EKR95] C.A. Ellis, K. Kedara, and G. Rozenberg. Dynamic change within workflow systems. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 10--21, Milpitas, California, August 1995. ACM.

[Ell79] C. Ellis. Information control nets: A mathematical model of office information flow. In *Proceedings of the 1979 ACM Conference on Simulation Measurement and Modelling of Computer Systems*, pages 225--239, 1979.

[Ell95] C.A. Ellis. Comunicação pessoal, 1995.

[EM93] C.A. Ellis and C. Maltzahn. Comparison of FlowPath, ActionWorkflow, and Viewstar. Technical report, University of Colorado at Boulder, Computer Science Dept., 1993.

[EM94] C.A. Ellis and C. Maltzahn. Collaboration with spreadsheets. *Journal of the Brazilian Computer Society*, 1(1):15--23, July 1994.

[EN93a] C.A. Ellis and G. Nutt. Modelling and analysis of coordination systems. Technical report, CU-CS-639-93, Department of Computer Science, University of Colorado at Boulder, 1993.

[EN94] R. Elmasri and S.B. Navathe. *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1994.

[EW94a] C.A. Ellis and J Wainer. Goal based models of collaboration. *Collaborative Computing*, 1(1), 1994.

[EW94b] C. Ellis and J. Wainer. A conceptual model of groupware. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, pages 79--88, 1994.

[For89] ForComment. *The ForComment System Users' manual*, 1989.

[FNO+93] G. Fischer and K. Nakakoji and J. Ostwald and G. Stahl snd T. Sumner. Embedding critics in design environments. *The knowledge engineering review*, 4(8), 1993.

[FTK95] G. Fitzpatrick, W.J. Tolone, and S.M. Kaplan. Work, locales and distributed social worlds. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work*, pages 1--16, Stockholm, Sweden, September 1995. Kluwer Academic Publishers. http://acsl.cs.edu/kaplan/papers/ecscw95.ps

[Gal77] J. R. Galbraith. *Organization Design.* Addison-Wesley, 1977.

[Hir86] R.A. Hirschheim. The effect of a priori views on the social implications of computing: The case of office automation. *ACM Computing Surveys*, 18(2):165--196, June 1986.

[HKE92] P. Hennessy, T. Kreifelts, and U. Ehrlich. Distributed work management: activity coordination within the eurocoop project. *Computer Communications*, 15(8):477--488, October 1992.

[Hsu93] M. Hsu (editor). Special issue on workflow and extended transaction systems. *IEEE Data Engineering Bulletin*, 16(2), June 1993.

[Hsu95] M. Hsu (editor). Special issue on workflow systems. *IEEE Data Engineering Bulletin*, 18(1), March 1995.

[Inc96a] Xerox Corporation. *InConcert: Workflow Software from XSoft.* http://www.xsoft.com/XSoft/DataSheets/InConcert.html.

[Inc96b] Xerox Corporation. *InConcert 3.0 - Adaptive Workflow Software for Continuous Process Improvement.* http://www.xsoft.com/XSoft/DataSheets/IC30.html.

[Ioc95] C. Iochpe. Improving requirements analysis through cscw. In *CYTED-RITOS International Workshop on Groupware*, Lisbon, Portugal, September 1995.

[Jen92] K. Jensen. *Coloured Petri Nets*. Springer Verlag, Berlin, 1992.

[Kar90] Karbe. Support of cooperative work by electronic circulation folders. In *Proceedings of the 1990 ACM Conference on Organizational Computing Systems (COOCS'90)*. ACM, 1990.

[KHK+91] T. Kreifelts, E. Hinrichs, K. Klein, P. Seufert, and G. Woetzel. Experiences with the domino office procedure system. In L. Bannon, M. Robinson, and K. Schmidt, editors, *Proceedings of the Second European Conference on Computer Supported Cooperative Work*, Amsterdam, September 1991.

[KHW93] T. Kreifelts, E. Hinrichs, and G. Woetzel. Sharing to-do lists with a distributed task manager. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work*, Milano, Italy, September 1993. Kluwer Academic Publishers.

[KLR+95] G. Kappel, P. Lang, S. Rausch-Schott, and W. Retschitzegger. Workflow management based on objects, rules, and roles. *Bulletin of the Technical Committee on Data Engineering*, 18(1):11--18, March 1995.

[KR95] M. Kamath and K. Ramamritham. Modeling, correctness & system issues in supporting advanced database applications using workflow management systems. Technical report 95-50, Department of Computer Science, University of Massachussets, 1995. http://www-ccs.cs.umass.edu/kamath/UM-CS-1995-050.ps.

[Kyn95] M. Kyng. Making representations work. *Communications of the ACM*, 38(9), September 1995.

[LMY88] K. Lai, T.W. Malone, and K. Yu. Object lens: a 'spreadsheet' for cooperative work. *ACM Transactions on Office Information Systems*, 6(4):332--353, October 1988.

[MCL+92] T. Malone, K. Crowston, J. Lee, and B. Pentland. Tools for inventing organizations: Toward a handbook of organizational processes. In *Proceedings of the 2nd IEEE Workshop on Enabling Tchnologies: Infrastructure for collaborative Enterprises (WET-ICE)*, pages 72--82, April 1993. ftp://pound.mit.edu/PUB/CCSWorking_Papers/CCSWP141.ps.

[Med92] P. Medina-Mora. Action workflow technology and applications for groupware. In D. Coleman, editor, *GroupWare'92*, 1992.

[MLF92] T.W. Malone, K. Lai, and C. Fry. Experiments with oval: A radically tailorable tool for cooperative work. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 289--297, Toronto, Canada, October 31 to November 4 1992. ACM Press. ftp://pound.mit.edu/PUB/CCSWorking_Papers/CCSWP181.ps.

[MS93] D.R. McCarthy and S.K. Sarin. Workflow and transactions in inconcert. *Bulletin of the Technical Committee on Data Engineering*, 16(2):53--56, June 1993.

[MWF93] R. Medina-Mora, H.K.T. Wong, and P. Flores. ActionWorkflow as the enterprise integration technology. *Bulletin of the Technical Committee on Data Engineering*, 16(2):49--52, June 1993.

[Oli93] J.L.Oliveira. Uma ferramenta gráfica para navegação e consulta em bancos de dados orientados a objetos. Master's thesis, Departamento de Ciência da Computação da Universidade Estadual de Campinas, 1993.

[OCM95] J. L. Oliveira, C. Q. Cunha, and G. C. Magalhães. Modelo de objetos para contrução de interfaces visuais dinâmicas. In *Proc. of the 9th Brasilian Symposium on Software Engineering*, pages 143--158, Recife, PE, Brasil, October 1995.

[OM95] J. L. Oliveira and C. B. Medeiros. A direct manipulation user interface for querying geographic databases. In *Proc. 2nd International Conference on Applications of Databases*, Sao Jose, USA, December 1995.

[Orl92] W.J. Orlikowski. Learning from notes: Organizational issues in groupware implementation. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work (CSCW'92)*. ACM, 1992.

[Pri93] W. Prinz. Tosca - providing organisational information to cscw application. In C. Simone G. De Michelis and K. Schmidt, editors, *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, pages 139--154, Milan, Italy, September 1993. Kluwer Academic Publishers, Dordrecht. ftp://ftp.gmd.de/gmd/cscw/TOSCA-ECSCW93.ps.gz.

[PZM+94] A. Pirotte, E. Zimanyi, D. Massart, and T. Yakusheva. Materialization: a powerful and ubiquitous abstraction pattern. In *Proceedings of the 20th VLDB Conference*, pages 630--641, Santiago, Chile, 1994.

[RB91] M. Robinson and L. Bannon. Questioning representations. In *Proceedings of 2nd European Conference on Computer Supported Cooperative Work*, pages 219--233, Amsterdam, Netherlands, September 1991.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[Rei82] W. Reisig. *Petri-nets*. Springer-Verlag, Berlin Heidelberg New York, 1982.

[Rei92] G.L. Rein. *organization Design Viewed as a Group Process Using Coordination Technology*. PhD thesis, Department of Information Systems, University of Texas at Austin, 1992.

[Rob93] M. Robinson. Design for unanticipated use... In C. Simkone G. de Michelis and K. Schmidt, editors, *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, pages 187--202, Milan, Italy, September 1993 . Kluwer Academic Publishers, Dordrecht.

[Rog94] Y. Rogers. Exploring obstacles: Integrating cscw in evolving organizations. In *Proceedings of CSCW'94*. ACM, 1994.

[Sac95] P. Sachs. Transforming work: Collaboration, learning and design. *Communications of the ACM*, 38(9):36--44, September 1995.

[Sea69] J.L. Searle. *Speech Acts*. Cambridge University Press, Cambridge, 1969.

[Sea79] J.L. Searle. *Expression and Meaning*. Cambridge University Press, Cambridge, 1979.

[SI95] K.D. Swenson and K. Irwin. Workflow technology: Tradeoffss for business process re-engineering. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, Milpitas, California, August 1995. ACM.

[SIM+94] K. D. Swenson et al. A business process environment supporting collaborative planning. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94)*. ACM, 1994. ftp://ftp.ossi.com/pub/regatta/ Schaerding.ps.

[Sin92] B. Singh. Interconnected roles (ir): a coordination model. Technical report, Micro-Eletronics and Computer Technology Corporation, 1992.

[SM89] D.M. Strong and S.M. Miller. Exception handling and quality control in office operations. Working Paper Number 89-16, Boston University, School of Management, Boston, MA, 1989.

[SMM+94] K. D. Swenson et al. Collaborative planning: Empowering the user in a process environment. *Collaborative Computing*, 1(1), 1994. ftp://ftp.ossi.com/pub/regatta/JournalCC.ps.

[SMS94] H. Saastamoinen, M. Markkanen, and V. Savolainen. Survey on exceptions in office information systems. Technical report, Department of Computer Science, University of Colorado at Boulder, 1994.

[SMS94] H. Saastamoinen. Exceptions: Three views and a taxonomy. Technical report, Department of Computer Science, University of Colorado at Boulder, 1994.

[Str93] A. Strauss. *Continual Permutations of Action*. Aldine de Gruyter, New York, 1993.

[Suc83] L.A. Suchman. Office procedure as practical action: Models of work and system design. *ACM Transactions on Office Information Systems*, 1(4):320--328, 1983.

[Suc87] L. A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge (UK), 1987.

[Suc93] L. A. Suchman. Do categories have politics? the language/action perspective reconsidered. In C. Simone G. de Michelis and K. Schmidt, editors, *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, 1993.

[Suc95] L. A. Suchman. Making work visible. *Communications of the ACM*, 38(9), September 1995.

[SW95] H. Saastamoinen and G.M. White. On handling exceptions. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 302--310, Milpitas, California, August 1995. ACM Press.

[TKF95] W.J. Tolone, S. Kaplan, and G. Fitzpatrick. Specifying dynamic support for collaborative work within worlds. In *Proceedings of the 1995 ACM Conference on Organizational Computing Systems (COOCS'95)*, pages 55--65, Milpitas, California, August 1995. ACM Press.

[WB96] J. Wainer and P. Barthelmess. Workcase-centric workflow model. Submetido ao NSF Workshop on Workflow and Process Automation, 1996.

[Win86] T. Winograd. A language/action perspective on the design of cooperative work. In *CSCW+86 Proceedings*. ACM, 1986.

[Win88] T. Winograd. Where the action is. *Byte*, pages 256A--258, December 1988.

[WMC95] Workflow Management Coalition. *Glossary - A Workflow Management Coalition Specification*. http://www.aiai.ed.ac.uk:80/WfMC/glossary.html.