Algoritmo para Recuperação de Falhas em Sistemas de Gerenciamento de *Workflow*

Adriano Fiad Farias[1,2], **Vinícius Cristiano de Almeida**[1,3], **Alexandre Fieno da Silva**[3]

[1]Universidade Federal de Uberlândia – Pós-graduação em Ciência da Computação
Campus Santa Mônica – 38400-902 – Uberlândia – MG – Brasil

[2]Centro Federal de Educação Tecnológica de Petrolina
Cx.Postal 178 – 56314-520 – Petrolina – PE – Brasil

[3]Faculdade do Noroeste de Minas – Tecnologia em Sistemas de Informação
Cx.Postal 201 – 38600-000 – Paracatu – MG – Brasil

{adrifiad,vinicius}@pos.facom.ufu.br

Resumo. *A recuperação de falhas em sistemas de workflow é um assunto muito discutido no meio científico e um grande desafio. Existem muitos trabalhos relacionados onde as abordagens do problema propõem soluções isoladas e desunificadas. Neste artigo temos como objetivos apresentar um estudo sobre recuperação de falhas em sistemas de gerenciamento de workflow (SGWF) e propor um algoritmo para tratamento do cancelamento das atividades correlacionadas por atividade com falha.*

1. Introdução

Atualmente sistemas de *workflow* se tornaram populares, pois auxiliam a resolução de problemas dos processos organizacionais. Essa popularização fez com que diversas empresas organizadas na forma de consórcios e associações sem fins lucrativos, conjuntamente com a comunidade acadêmica se esforçassem na padronização desses sistemas.

Workflow pode ser definido como qualquer tarefa executada em série ou em paralelo por dois ou mais membros de um grupo de trabalho, visando um objetivo comum. *Workflow* é usado como um sinônimo para "processo de negócio". Segundo a *WorkFlow Management Coalition* (WfMC), *workflow* é a automação total ou parcial de um processo de negócio, durante a qual, documentos, informações ou tarefas são passados de um participante para outro para a realização de alguma ação, de acordo com um conjunto de regras procedimentais. Aplicado dentro do contexto de uma estrutura organizacional, define atividades funcionais e relações. Essas atividades podem ser humanas (reuniões e entrevistas) ou automatizadas (a impressão de um documento).

A automatização do processo de negócio se faz através de três elementos básicos: papéis, regras e rotas, através desses elementos a tecnologia *workflow* possibilita o controle geral de um processo e suas atividades em termos de tempo e movimento, sendo assim possível rastrear "o quê" está atrasado e o "porquê" do atraso. Para que um *workflow* possa ser definido e executado, com participantes realizando atividades de negócios, é necessário um sistema de gerenciamento de *workflow* (SGWF).

A WfMC entende sistemas de gerenciamento de *workflow* como sistemas que definem, executam e gerenciam completamente *workflows* através da execução de um software, cuja ordem de execução é dirigida por uma representação lógica e computadorizada

de um *workflow*. Uma das contribuições mais importantes do consórcio WfMC foi a padronização de diversos conceitos relativos a *workflows*. A Figura 1 apresenta estes principais conceitos e seus relacionamentos.

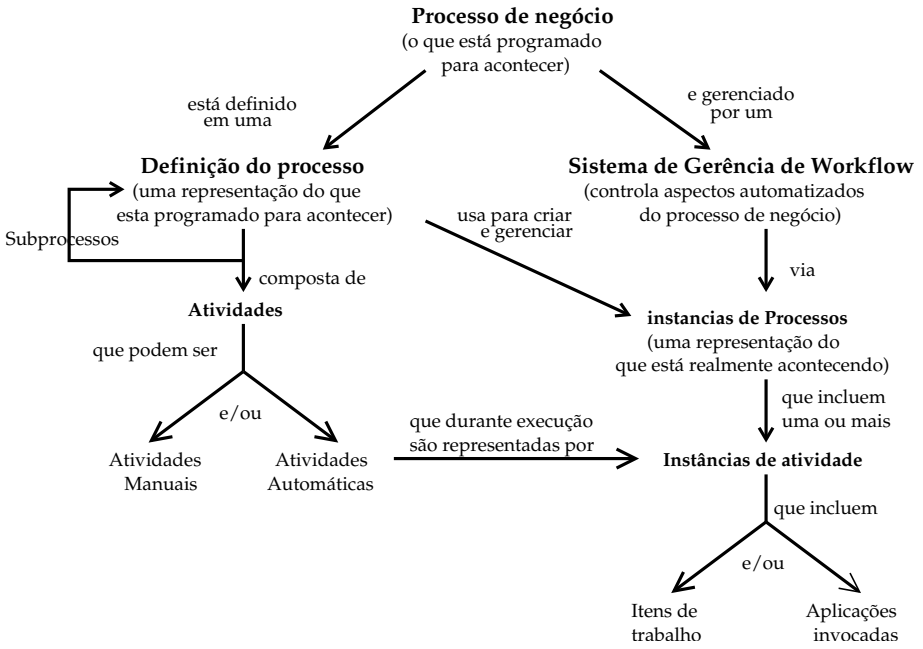

Figura 1. Conceitos relativos a sistemas de *workflow* padronizados pelo WfMC

Segundo [Georgakopoulos et al. 1995] a gerência de *workflow* envolve desde a modelagem dos processos até a sincronização das atividades e dos participantes que realizam os processos. São destacadas as seguintes etapas na gerência de *workflow*: (i) modelagem do processo e especificação; (ii) reengenharia do processo, e (iii) implementação e automação do *workflow*.

O importante é que os sistemas de *workflow* garantam que a execução nunca seja interrompida, independentemente da falha ocorrida. Falha é um evento que causa erros. De acordo com [Heinl et al. 1999] muitas falhas ou exceções, ocorrem nos SGWF, tornando impossível que o sistema identifique essas falhas. Nestas situações, apenas a intervenção humana, de alguém que tenha uma perspectiva global do sistema, pode identificar e definir o mecanismo de tratamento apropriado. Segundo [Worah and Sheth 1997] há três categorias de erros ocasionados por falhas associadas a sistemas de gerência de *workflow*: (i) erros de infra-estrutura; (ii) erros de sistema e (iii) erros de aplicações e usuários.

[Dayal et al. 1991] reconhece uma estrutura fixa de controle e políticas rígidas de compensação. Regras são desenvolvidas para desacoplar a detecção e o tratamento das exceções da execução do fluxo de trabalho aumentando a flexibilidade da abordagem. [Casati et al. 1999] descreve um sistema para tratamento de exceções esperadas baseado em regras desenvolvidas por Dayal, afirmando que as exceções não esperadas devem ser tratadas por humanos uma vez que não foram previstas durante a fase de modelagem.

Em [Fisteus 2005] foi proposto uma arquitetura para análise dos padrões de *workflow*, entre eles o padrão 19 responsável pelo cancelamento das atividades, sendo proposto uma codificação formal. Dentro desta perspectiva este trabalho apresenta um estudo sobre

recuperação de falhas em sistemas de gerenciamento de *workflow* e propõe um algoritmo para tratamento do cancelamento das atividades correlacionadas a atividade com falha.

Para tanto este artigo está dividido da seguinte forma: na seção 2 é apresenta uma visão dos modelos de *workflow*. O tratamento das falhas e exceções são abordados na seção 3. As formas de recuperação de *workflow* em caso de falhas são apresentadas na seção 4. Na seção 5 é apresentado o algoritmo proposto por este trabalho. As conclusões são apresentadas na seção 6 e por fim as referências bibliográficas.

2. Modelos de Workflow

O termo *workflow* transacional foi introduzido em 1993 enfatizando a relevância das propriedades transacionais nos *workflows*. Propriedades essas, necessárias para especificar critérios de coordenação, corretude, consistência de dados, confiabilidade e suporte a falhas [Worah and Sheth 1997].

Dentre as propriedades apresentadas acima, podem ser citadas aqueles relativas ao controle de concorrência sobre ítens de dados vinculados a diversas atividades ou *subworkflows* distintos, à recuperação em caso de falhas e a coordenação das atividades envolvidas em um determinado *workflow*.

O primeiro grande passo na evolução dos modelos transacionais foi exatamente a extensão da estrutura das transações planas para uma estrutura hierárquica, de múltiplos níveis, cujo modelo foi chamado de Modelo de Transações Aninhadas. Neste modelo, (a) uma transação-filha apenas começa depois que sua transação-pai tiver começado; (b) uma transação-pai apenas termina depois que todas as suas transações-filhas forem terminadas e (c) se uma transação-pai é abortada, todas as suas transações-filhas também são.

Modelar um *workflow* como uma transação estendida significa que as subtransações correspondem às tarefas do *workflow* ou a subworkflows e a estrutura de execução da transação estendida corresponde ao fluxo de controle do *workflow*. Transações aninhadas, Sagas, transações flexíveis, dentre outros, são alguns dos modelos estendidos encontrados na literatura. Um breve resumo destes modelos é apresentado em [Rusinkiewicz and Sheth 1995], [Worah and Sheth 1997].

3. Taxionomia de Tratamento de Exceções

As taxonomias existentes que abordam o tratamento de exceções na literatura podem ser vistas em duas distintas perspectivas: sistêmica e organizacional. A perspectiva sistêmica de [Eder and Liebhart 1998] caracteriza-se por falhas e exceções numa única dimensão, composta por dois tipos de falhas (falhas elementares e nas aplicações) e dois tipos de exceções (exceções esperadas e não esperadas).

As falhas elementares apresentam faltas em nível do sistema de suporte ao SGWF (sistema operacional, SGBD ou faltas na rede). As falhas nas aplicações, apresentam falhas na implementação das tarefas (entrada inesperada de dados);

As exceções esperadas são eventos ou situações que podem ser previstas durante a fase de modelagem mas, que não correspondem ao comportamento "normal" de um processo. Este tipo de exceção pode ocorrer com freqüência e originar uma quantidade de trabalho significativo em seu tratamento. Devem ser previstos mecanismos para o tratamento destas situações uma vez que podem ocorrer com freqüência [Eder and Liebhart 1998].

[Casati et al. 1999] identifica quatro classes de exceções esperadas de acordo com os eventos que as originam: (a) exceções de fluxos de trabalho, (b) exceções dos dados, (c) exceções temporais e (d) exceções externas.

As exceções não esperadas ocorrem quando a semântica do processo não é corretamente modelada no sistema. São exemplos, mudanças nas regras devido a alterações legislativas, mudanças estruturais que envolvem a organização ou alteração no processamento de uma compra efetuada por um cliente especial.

Ainda em [Casati et al. 1999], as exceções não esperadas resultam de inconsistências entre a modelagem do processo no fluxo de trabalho e a execução efetiva. Essas exceções resultam de falhas de projeto ou projeto incompleto, melhoramentos ou alterações no negócio ou a necessidades de satisfação dos clientes não previstas durante a fase de modelagem. Este tipo de exceções não esperadas é freqüente em ambientes dinâmicos ou complexos. Essas exceções podem obrigar a interrupção da execução automática do processo e a intervenção de um operador [Heinl et al. 1999].

Em situações onde este tipo de exceção não esperada ocorre com freqüência, deve ser considerada a reformulação do modelo do fluxo de trabalho, a adoção de outras tecnologias baseadas em trabalho cooperativo ou sistemas de fluxos de trabalho adaptativos.

4. Recuperação em Caso de Falhas

Existem várias classificações para falhas na literatura. Normalmente, essas classificações agrupam as falhas em físicas e humanas. Falhas físicas referem a falhas de componentes de hardware e falhas humanas compreendem falhas de projeto, decorrentes das fases de desenvolvimento do software, falhas de interação podem ser acidentais ou intencionais.

Caso o estado errôneo do sistema não seja tratado em um tempo determinado, ocorrerá um defeito, que se manifestará pela não execução ou mudança indesejada no serviço especificado. Como pode ser visto na Figura 2, além de indicar um estado errôneo, a ocorrência de um defeito realimenta o ciclo e pode gerar novas falhas. Esse fenômeno é conhecido como propagação da falha e deve ser contido através do seu confinamento.

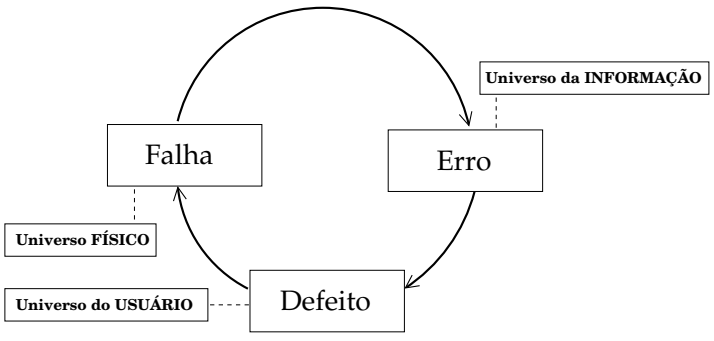

Figura 2. Ciclo entre a ocorrência de falhas, erros e defeitos (Brito:05)

As falhas também podem ser classificadas segundo a sua duração. Nessa perspectiva, as falhas são distribuídas em três grupos: (i) transientes, quando existe a chance das falhas ocorrerem mais de uma vez mas não obrigatoriamente, por exemplo, a invasão do sistema por um hacker; (ii) intermitente, quando a ocorrência da falha se repete, mas não necessariamente em períodos definidos, por exemplo, a manifestação de um vírus pre-

sente no sistema; e (iii) permanentes, que são caracterizadas pela sua presença constante no sistema, por exemplo, falhas de especificação e implementação [Brito 2005] .

Segundo [Worah and Sheth 1997] falhas associadas ocasionam três categorias de erros a SGWF: (i) erros de infra-estrutura - falhas de comunicação ou hardware; (ii) erros de sistema - falha no próprio software do sistema de *workflow* e (iii) erros de aplicações e usuários - decorrente da execução de determinada tarefa por parte dos mesmos.

Se o erro é de sistema e ocorre no banco de dados onde estão contidas as informações dos workflows, o mecanismo de replicação, se presente, pode resolver o problema. Neste caso, quando os dados necessários para a execução da tarefa estiverem locais ao dispositivo em que ela está sendo executada, a execução não é interrompida.

No contexto de recuperação em caso de falhas, algumas alternativas podem ser utilizadas com o intuito de evitar que a execução seja interrompida. Por exemplo, quando uma tarefa não puder ser executada, diversas tentativas podem ser feitas até que se consiga executá-la (como no caso da falta de recursos) ou tarefas alternativas podem ser executadas (quando um certo número de tentativas falhou) para evitar que a tarefa falhe. Se uma tarefa alternativa for executada no lugar da que tinha falhado, considera-se que a falha foi resolvida. No entanto, esta abordagem de utilização de tarefas alternativas ou novas tentativas não ocorre em grande parte dos sistemas de workflow. Para que a recuperação seja possível no nível de tarefas e não apenas no nível de workflows, é necessário que toda a informação de execução do workflow, inclusive aquelas relativas a cada tarefa individual, sejam armazenadas de modo persistente.

Se a execução de uma instância de *workflow* é de fato abortada, ações compensatórias devem ser executadas pela máquina de execução para desfazer os efeitos semânticos das operações, sempre que possível. Estas ações não levam, necessariamente, o banco de dados que controla a execução do *workflow* para o estado válido anterior à falha, mas sim o leva para um estado válido e semanticamente próximo. Isso porque muitas vezes os efeitos transacionais das tarefas de um *workflow* não podem ser desfeitos.

Existem basicamente duas maneiras possíveis de se recuperar uma instância de *workflow* que falhou: através de *backward recovery* e através de *forward recovery*. Na abordagem de *backward recovery* a instância de *workflow* é retornada, através de ações compensatórias, para o estado consistente que existia antes da transação que representa a instância de *workflow* abortada (que falhou). Na abordagem de *forward recovery* a instância que falhou volta para um estado válido e continua a sua execução a partir do ponto em que havia falhado.

Mecanismos de pontos de checagem podem ser úteis tornando o processo de recuperação eficiente, salvando dados de tempos em tempos. Mecanismos de recuperação são sempre intimamente ligados à infra-estrutura do sistema, à sua arquitetura, à natureza dos executores, aos tipos de tarefas e da natureza da própria aplicação de *workflow*. Assim um único mecanismo de recuperação de falhas poderá não ser capaz de resolver problemas de vários sistemas semanticamente distintos.

5. Proposta Algorítmica

No estudo de [Eder and Liebhart 1998], [Klein and Dellarocas 2000] e [Dayal et al. 1991], prevêem que as possíveis causas de falhas e exceções são difí-

ceis ou mesmo impossíveis de serem tratadas unificadamente. Tornando os sistemas mais complexos e difíceis de gerir. Assim, preparar sistemas para lidarem com estas situações durante a fase de execução é um fator crítico de sucesso na implementação de SGWF.

As organizações são ambientes complexos e a aproximação tradicional baseada apenas na integridade e consistência dos dados não constitui um suporte suficientemente sólido. Apesar das limitações reconhecidas, o sistema de suporte do SGWF deve ser suficientemente robusto e suportar a integridade e consistência dos dados relevantes.

Como mencionado anteriormente, um esforço significativo foi investido no tratamento das falhas e exceções utilizando técnicas de sistemas de SGBD. No entanto, a semântica das tarefas nos SGWF excede largamente os modelos transacionais de SGBD. Por exemplo, em um *processo_a* em execução, a *atividade_n* desse processo (i.e, telefonema para um cliente) falha (por este não atender) não é necessário fazer nada de momento; é esperado por um tempo pré-determinado, que não exceda o tempo de execução desta atividade, para repetir a *atividade_n* até que esta seja concluída.

A semântica do tratamento de erros em sistemas tradicionais é muito rígida para os SGWF [Worah and Sheth 1997]. A integridade do sistema é garantida assegurando que caso a *atividade_n* seja cancelada, as *atividades_n+m* que instanciarem as alterações provisórias sejam informadas e reajam em concordância. Por outro lado, políticas rígidas de compensação implementam as atividades que devem ser efetuadas para voltar a colocar o sistema em um estado coerente sempre que uma atividade é cancelada.

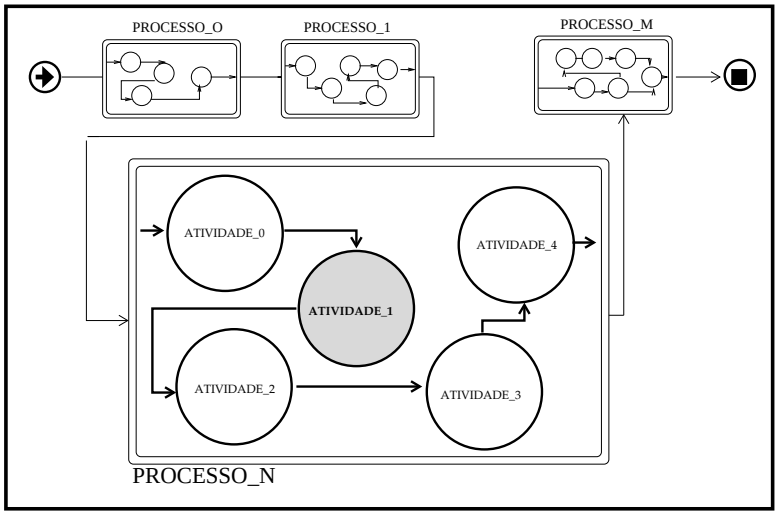

Figura 3. Processo de cancelamento de atividade.

Considere o *workflow* com ocorrência de falha na execução de atividades relacionadas a determinado processo. Se o processo possuir atividades correlacionadas entre si, como proceder? Nesse momento é necessário um algoritmo que trate o cancelamento das atividades correlacionadas a atividade com falha, implementando mecanismos que assegurem o sistema a retornar para um estado válido ou o mais próximo possível.

Por exemplo, uma *atividade_1* de um *Processo_N* falha. Todas as atividades relacionadas a este processo são canceladas. O sistema de workflow, deve implementar métodos para que a execução retorno a um estado válido, no caso o processo anterior (*processo_1* na Figura 3). Partindo dessa premissa, propomos algoritmicamente na Figura 4,

solução frente ao tratamento desse tipo de falha ocorrida.

O algoritmo está posicionado da seguinte forma *[Processo_0,..., Processo_N,..., Processo_M]* e a atividade do processo *[A_0,..., A_n,..., A_m]*. Considere a modelagem deste algoritmo mediante a utilização de um atributo adicional booleano, cujo valor inicial deve ser falso. O algoritmo como se pode observar, verifica se uma atividade anterior a atividade a ser executada foi cancelada ou não.

```
PROCESSO_N

1   Criar Tipo Cancel
2           cancelamento : booleano
3
4   Entidade A_n : cancel
5
6   VERIFICA_TRANSAÇÃO A_n−1
7       SE  ((estad.A_n != cancelado) e (estado.A_n−1 = cancelado))
8                   ENTÃO (A_n.cancel = true)
9
10  INÍCIO_TRANSAÇÃO A_n
11      SE ((estado.A_n = parado) e (estado.A_n−1 = finalizado)
12        e (A_n.cancel = false))
13                ENTÃO Início execução A_n
14                        SE término execução esperada (estado.A_n = finalizado)
15                        SE término execução inesperada (estado.A_n = cancelado)
16
17  CANCELAMENTO_TRANSAÇÃO A_n
18      SE ((estado.A_n != cancelado) e (estado.A_n = finalizado)
19        e (A_n.cancel = true))
20                ENTÃO Início cancelamento A_n
```

Figura 4. Algoritmo de tratamento de falhas das atividades.

Se o estado da transação atual for diferente de cancelado e o estado da atividade anterior foi cancelado, então a atividade atual é cancelada (Figura 4 - linhas 7-8). Após a *verificação_transação A_n-1*, o algoritmo executa o início da transação *A_n*.

Se o estado da transação *A_n* encontra-se parado e o estado da transação *A_n-1* finalizado e o valor booleano de *A_n*.cancel for igual a false, então se dá início a execução da atividade *A_n*. É monitorado se o término da execução é esperado ou inesperado, atribuindo ao estado da atividade *A_n*, finalizado ou cancelado, respectivamente.

Após o teste para início da execução da transação duas possibilidades se fazem possíveis. Se a condição das linhas 11-12 forem verdadeiras, é dado início a transação, podendo terminar de forma esperada ou inesperada. Se a condição for falsa, o algoritmo verifica o cancelamento da transação da atividade *A_n*.

Para a verificação do cancelamento da transação *A_n*, são observadas algumas variáveis como o estado da transação *A_n* e o valor booleano de *A_n*.cancel. Se o estado de *A_n* for igual a cancelado ou parado e o estado da variável *A_n*.cancel for true, então se dá início ao cancelamento da atividade. É interessante lembrar que está proposta foi baseada no motor de *workflow* YAWL, onde em sua arquitetura implementa métodos de recuperação de falhas *forward recovery*.

Apesar dos esforços para tratar automaticamente as exceções, parece impossível incluir semântica específica de uma tarefa numa plataforma de recuperação genérica. Uma vez que o comportamento das tarefas é ortogonal ao processo de fluxo de trabalho.

6. Conclusões

Este trabalho apresenta uma proposta para implementação de um algoritmo que trata o cancelamento das atividades subsequentes a atividade que ocorreu falha. Constatamos, que o assunto é amplo e muito discutido no meio científico. Existem vários trabalhos relacionados, onde as abordagens do problema propõem soluções isoladas e desunificadas, tornando o tratamento de falhas em SGWF algo desafiador. Esse algoritmo proposto é capaz de fazer o cancelamento de atividades que ocasionem falhas dentro de um processo em execução, adotando política preventiva de erros.

Como futuras extensões a este trabalhos será efetuado um aprofundamento no estudo, para implementação do algoritmo proposto junto ao motor *workflow* YAWL. Aplicação de técnicas de detecção de falhas em conjunto com o algoritmo proposto. A utilização de técnicas de mineração de dados e unificação de todas as possíveis soluções descritas na literatura para tratamento e recuperação de falhas em sistemas de workflow.

Referências

Brito, P. H. (2005). Um método para modelagem de exceções em desenvolvimento baseado em componentes. *Dissertação de mestrado - UNICAMP*.

Casati, F., Ceri, S., Paraboschi, S., and Pozzi, G. (1999). Specification and implementation of exceptions in workflow management systems. *ACM Trans. Database Syst.*, 24(3):405–451.

Dayal, U., Hsu, M., and Ladin, R. (1991). A transactional model for long-running activities. In *VLDB '91: Proceedings of the 17th International Conference on Very Large Data Bases*, pages 113–122, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Eder, J. and Liebhart, W. (1998). Contributions to exception handler in workflow management. Int. Conf. on Extended Database Technology (EDBT'98).

Fisteus, J. A. (2005). Definición de un modelo para la verificación formal de procesos de negocio. Tesis Doctoral. Universidad Carlos III de Madrid. Departamento de Ingeniería Telemática. Leganés,Spain.

Georgakopoulos, D., Hornick, M. F., and Sheth, A. P. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153.

Heinl, P., Horn, S., Jablonski, S., Neeb, J., Stein, K., and Teschke, M. (1999). A comprehensive approach to flexibility in workflow management systems. In *WACC '99: Proceedings of the international joint conference on Work activities coordination and collaboration*, pages 79–88, New York, NY, USA. ACM Press.

Klein, M. and Dellarocas, C. (2000). A knowledge-based approach to handling exceptions in workflow systems. *Computer Supported Cooperative Work*, 9(3/4):399–412.

Rusinkiewicz, M. and Sheth, A. P. (1995). Specification and execution of transactional workflows. In *Modern Database Systems: The Object Model, Interoperability, and Beyond.*, pages 592–620.

Worah, D. and Sheth, A. P. (1997). Transactions in transactional workflows. In *Advanced Transaction Models and Architectures*, pages 3–34.