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

IGOR STEINMACHER

Estudo e Análise de Desempenho da Antecipação de Tarefas em Sistemas Gerenciadores de *Workflow*

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. José Valdeni de Lima
Orientador

Porto Alegre, março de 2005.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Steinmacher, Igor
Estudo e Análise de Desempenho da Antecipação de Tarefas em Sistemas Gerenciadores de Workflow/ Igor Steinmacher. – Porto Alegre: Programa de Pós-Graduação em Computação, 2005.

102 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2005. Orientador: José Valdeni de Lima.

1. *Workflow*. 2. Flexibilidade 3. Antecipação 4. Simulação I Lima, José Valdeni de. II Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. José Carlos Ferraz Hennemann

Vice-Reitor de Ensino: Prof. Pedro Cezar Dutra Fonseca

Pró-Reitora de Pós-Graduação: Profª. Valquíria Link Bassani

Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux

Coordenador do PPGC: Prof. Flávio Rech Wagner

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus pelo dom da vida e pela oportunidade de dar mais este passo em minha vida. Agradeço também aos meus pais: meu pai, João, e minha mãe, Maria, amo vocês. Obrigado pelo amor, conforto e por terem me servido de exemplo. Um forte abraço e um muito obrigado ao meu orientador José Valdeni de Lima, que me "agüentou" durante estes quase dois anos, aconselhando e guiando meu trabalho. Estendo este agradecimento ao PPGC/UFRGS, seus professores e funcionários. Agradeço ainda à galera do laboratório do grupo SIGHA, em especial: Tiago Telecken, Níccholas Vidal e Carlos Morais. Agradeço ainda aos amigos que fiz no mestrado e me ajudaram no andamento do trabalho: Robson Cordeiro e Ana Vitória.

Obrigado aos meus entes mais próximos que participaram desta minha caminhada. Cláudia, obrigado pelo apoio psicológico. Ivan, valeu pelas consultas à distância. Denerval, obrigado pelas dicas, idéias e apoio técnico. Agradeço à minha querida namorada. Dani, obrigado por me esperar todo este tempo distante, pela confiança e pelo carinho que teve comigo, amo você. Um viva aos queridos amigos Fábio e Paulo que tiveram o prazer de dividir o apartamento comigo nestes dois anos. Ganhei dois irmãos. Carolina Beal, valeu pela força no início, pela cama, escrivaninha, armário, colchão e outras coisas que herdei do seu retorno a Maringá.

A todos os amigos e colegas que fiz em Porto Alegre. Saibam que, mesmo que eu volte para a terra natal, vocês continuarão sendo meus amigos. Galera do Charrua Rugby, valeu pela brutalidade e companheirismo. Pessoal do Rotaract Bom Fim, super abraço pra vocês. Enfim, um muito obrigado ao povo gaúcho, que me acolheu *tri bem* e me ensinou o estilo de vida *gaudério*.

Ao pessoal de Maringá que deixou saudades e sempre que possível estavam em contato, me visitando e fazendo festas.

Ao álcool etílico, em especial àquele contido na cerveja, que me acompanhou em minhas comemorações e me consolou sempre que precisei. Estendo este agradecimento ao Shamrock, um *pub* de nível.

À Gol Linhas Aéreas Inteligentes pelas passagens baratas a valer!

Agradeço à música de boa qualidade feita pelo mundo afora, que me acompanhou durante toda a concepção deste trabalho.

Ao sofrimento e dor causados por este trabalho, pois deles tirei lições valiosas.

Finalmente, agradeço ao CNPq que forneceu suporte financeiro, sem o qual não seria possível a realização deste trabalho.

Não sei onde eu tô indo,
mas sei que tô no meu caminho.
Raul Santos Seixas

SUMÁRIO

LISTA DE ABREVIATURAS 7

LISTA DE FIGURAS 8

LISTA DE TABELAS 10

RESUMO 11

ABSTRACT 12

1 INTRODUÇÃO 13

1.1 Organização do Trabalho 14

2 CONCEITOS E TECNOLOGIAS UTILIZADAS 15

2.1 *Workflow* 15
2.1.1 *Workflow* - Conceito 15
2.1.2 Conceitos Básicos 15
2.1.3 Fluxo de Controle 18
2.1.4 Fluxo de Dados 19
2.1.5 Arquitetura de um Sistema Gerenciador de *Workflow* 20
2.1.6 Modelo de Referência para SGWfs 22
2.1.7 XPDL 23
2.2 Mecanismos de Percepção 24
2.2.1 Classificação dos tipos de elementos de percepção 25
2.3 Considerações Finais 28

3 FLEXIBLIDADE EM SISTEMAS DE *WORKFLOW* 29

3.1 Flexibilidade *a priori* 29
3.2 Flexibilidade *a posteriori* 34
3.3 Uma abordagem mais abrangente 37
3.4 Considerações Finais 39

4 ANTECIPAÇÃO DE TAREFAS 41

4.1 Antecipação Gerenciada *a priori* 42
4.2 Antecipação Gerenciada *a posteriori* 45
4.2.1 Identificação das Tarefas Durante a Definição de Processos 45
4.2.2 Flexibilizando o fluxo de controle 46
4.2.3 Fluxo de Dados 48
4.2.4 Modificando a Máquina de Execução 50

4.3 Percepção como apoio à Antecipação 57
4.4 Considerações Finais..... 60

5 SIMULAÇÕES 62

5.1 Cenário Utilizado..... 62
5.1.1 Execução Utilizando a Abordagem Tradicional 63
5.1.2 Execução Utilizando Antecipação de Tarefas 64
5.1.3 Comparando as duas ocorrências..... 66
5.2 Simulações realizadas 68
5.2.1 Resultados..... 69
5.3 Considerações Finais..... 76

6 IMPLEMENTAÇÕES 78

6.1 Simulador 78
6.1.1 Ferramenta de definição 79
6.1.2 XSLT..... 80
6.1.3 SGBD 81
6.1.4 Motor..... 82
6.1.5 Interface com o Usuário 83
6.2 Ferramenta de definição de processos *Amaya Workflow* 86
6.3 Máquina de *Workflow* 88

7 CONCLUSÕES 90

7.1 Trabalhos Futuros 92
7.2 Publicações 92

REFERÊNCIAS.....94

APÊNDICE A AUMENTANDO A COOPERAÇÃO ATRAVÉS DA ANTECIPAÇÃO DE TAREFAS.....99

A.1 Mecanismos de trocas de resultados entre tarefas 99
A.2 Estratégia de atualização dos elementos de dados – *push x pull* 101

LISTA DE ABREVIATURAS

AW	Amaya *Workflow*
CEMT	Conception d'un Environnement Éditorial Multimédia coopératif pour le web avec Technologie de *workflow*
ECA	Evento Condição Ação
FCFS	First to Come, First to Serve
HTML	Hypertext Markup Language
PERT	Program Evaluation and Review Technique
PHP	PHP: Hypertext Preprocessor
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SGWf	Sistema Gerenciador de *Workflow*
SQL	Structured Query Language
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
WAPI	*Workflow* Application Programming Interface
WfMC	*Workflow* Management Coalition
WIDE	*Workflow* on Intelligent Distributed database Environment
WWW	World Wide Web
XML	eXtensible Markup Language
XPDL	XML Process Definition Language
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language – Transformation

LISTA DE FIGURAS

Figura 2.1: Relacionamento entre conceitos básicos de *workflow* 16
Figura 2.2: Exemplo básico de definição de processo 17
Figura 2.3: Exemplo de tarefas seguindo um fluxo seqüencial 19
Figura 2.4: Tarefas obedecendo a um fluxo paralelo 19
Figura 2.5: Utilização de elementos *OR-join* e *OR-split* provendo condicionalidade 19
Figura 2.6: Fluxo iterativo, que pode ser definido através da condição de saída da tarefa 19
Figura 2.7: Processo explicitando conectores do fluxo de dados e do fluxo de controle 20
Figura 2.8: Estrutura Básica de um *workflow* proposta pela WfMC 21
Figura 2.9: Arquitetura detalhada de um sistema de *workflow* 22
Figura 2.10: Modelo de Referência WfMC 23
Figura 3.1: O uso do operador Coo 30
Figura 3.2: Inserindo (a) flexibilidade por seleção e (b) flexibilidade por adaptação 39
Figura 4.1: Exemplo básico do andamento de um processo utilizando antecipação de uma tarefa 42
Figura 4.2: Antecipação utilizando modelo tradicional de definição de processos utilizando menor granularidade na modelagem das tarefas 43
Figura 4.3: Tipos de dependência entre tarefas previstas 48
Figura 4.4: Diagrama de Transição de Estados proposto para os elementos de dados 49
Figura 4.5: Diagrama de Estados proposto pela WfMC 51
Figura 4.6: Diagrama de Estados para as Instâncias de Tarefas prevendo a antecipação 53
Figura 4.7: Especialização do Estado `nãoPronta` prevendo as condições de disparo 55
Figura 5.1: Processo criado para simulação 63
Figura 5.2: Processo executado utilizando a abordagem tradicional 64
Figura 5.3: Processo executado utilizando a antecipação de tarefas 66
Figura 5.4: Execução do processo de maneira tradicional 67
Figura 5.5: Execução do processo com antecipação de tarefas 67
Figura 5.6: Tempo Médio de execução dos processos 71
Figura 5.7: Tempo Médio de execução dos processos com valores mapeados para horas 72
Figura 5.8: Tempo Médio de execução dos processos levando em conta o início da antecipação e o esforço adicional 74

Figura 5.9: Tempo Médio de execução dos processos levando em conta o início da antecipação e o esforço adicional com valores mapeados para horas 75
Figura 5.10: Tempo Médio de execução dos processos com carga de trabalho 76
Figura 6.1: Arquitetura do protótipo de simulação implementado 79
Figura 6.2: Trecho de uma definição de processo em XPDL explicitando os atributos estendidos durMin e durMax 80
Figura 6.3: Estrutura do banco de dados que permitiu a importação dos dados 81
Figura 6.4: Interface inicial do protótipo do simulador implementado 83
Figura 6.5: Opções de simulação previstas no protótipo implementado 84
Figura 6.6: Acesso progressivo aos resultados da simulação 86
Figura 6.7: Interface da Ferramenta Amaya *Workflow* modificada para modelar tarefas antecipáveis 87
Figura 6.8: Trecho de definição de processos explicitando o atributo estendido criado para representar a antecipação de uma tarefa 87
Figura 6.9: Interface com o usuário da máquina de *workflow* alterada para suportar a antecipação de tarefas 89
Figura A.1: Possíveis arquiteturas para distribuição dos dados 99
Figura A.2: Arquitetura com dois níveis de repositório para possibilitar a troca de resultados entre tarefas 101

LISTA DE TABELAS

Tabela 3.1: Relacionamentos de precedência propostos no PROMENADE 31
Tabela 4.1: Combinação das condições de disparo e o estado que as representa..... 56
Tabela 4.2: Tipos de percepção e sua importância para o suporte à antecipação de tarefas 59
Tabela 5.1: Duração Mínima, Máxima e Média das tarefas..... 63
Tabela 5.2: Comportamento das tarefas durante a execução do processo de maneira tradicional..... 64
Tabela 5.3: Comportamento das tarefas durante a execução do processo com antecipação 65
Tabela 5.4: Comparativo entre a execução com antecipação e sem antecipação..... 67
Tabela 5.5: Duração Mínima, Máxima e Média para as tarefas – em dias e em horas. 69
Tabela 5.6: Tempo médio de execução dos processos variando o início da antecipação 70
Tabela 5.7: Tempo médio de execução das tarefas (em horas) variando o início da antecipação 71
Tabela 5.8: Tempo médio de execução das tarefas variando o início da antecipação e o esforço adicional 73
Tabela 5.9: Tempo médio de execução das tarefas (em horas) variando o início da antecipação e o esforço adicional 74
Tabela 5.10: Tempo médio de execução das tarefas variando o início da antecipação com *workload* 75
Tabela 6.1: Informações importadas dos arquivos XPDL para permitir a simulação ... 81

RESUMO

Sistemas Gerenciadores de *Workflow* (SGWf) têm atraído muita atenção nos últimos anos, por serem baseados em um modelo simples que permite definir, executar e monitorar a execução de processos. Mas este esquema é um tanto quanto rígido e inflexível, apresentando alguns problemas quando da necessidade de sua utilização no controle de processos com características cooperativas, em que o comportamento humano deve ser levado em conta. Este tipo de processo deve apresentar uma certa flexibilidade de forma a possibilitar a troca de resultados entre tarefas, acabando com a dependência *fim-início* existente entre tarefas seqüenciais nos SGWfs tradicionais. A fim de possibilitar este tipo de comportamento, surge a *antecipação de tarefas*, que tem por objetivo oferecer meios de flexibilizar o fluxo de controle e o fluxo de dados para permitir que tarefas possam ser disparadas antes mesmo da conclusão de suas antecessoras. A antecipação implica no aumento da cooperação entre tarefas e em melhor desempenho com relação ao tempo de execução dos processos. O aumento na cooperação está relacionado diretamente à troca de resultados, permitindo que duas ou mais tarefas trabalhem sobre instâncias de um determinado dado que ainda está sendo produzido. O melhor desempenho deve-se ao paralelismo (parcial) criado entre tarefas que deveriam executar de maneira estritamente seqüencial, devido ao disparo das tarefas antes do previsto. Este trabalho então, apresenta um estudo das abordagens de flexibilização de SGWfs, visando encontrar maneiras de prover antecipação de tarefas. O principal objetivo do trabalho é verificar a melhoria de desempenho em termos de tempo de execução dos processos que se pode obter utilizando a antecipação de tarefas. Para tal, foi desenvolvido um simulador que suporta a antecipação de tarefas e foram conduzidas simulações, que mostraram os ganhos e alguns padrões de comportamento da utilização da antecipação de tarefas. Para possibilitar o uso da antecipação em processos reais foram realizadas alterações na ferramenta de definição de processos *Amaya Workflow* e na máquina de *workflow* do projeto CEMT.

Palavras-chave: *workflow*, antecipação, flexibilização, cooperação, simulação.

Task Anticipation in Workflow Management Systems: a Study and Performance Analysis

ABSTRACT

Workflow Management Systems have attracted much attention recently because of their simple yet powerful model for defining, executing and monitoring processes. However, traditional Workflow Management Systems are too rigid, posing problems when cooperation and human behavior are important factors. Under these circumstances, there is a need for a relaxation of the end-begin dependency. Task anticipation can provide this relaxation. Task anticipation consists in relaxing the flow of control and data information among tasks, allowing the exchange of results among sequential tasks even before the conclusion of the former task. This anticipation promotes a better cooperation and augments the performance in terms of process execution time. The level of cooperation is directly related to the exchange rate of results, as two or more task can work with a same instance of a giving data while it is still being produced. Meanwhile, the performance is directly related to the level of parallelism created among tasks, as two or more tasks can be executed at some degree of parallelism even though they were initially defined to execute in a strictly sequential manner. The main goal of this work was to study mechanisms of task anticipation in Workflow Management Systems. In terms of bringing a better understanding of the advantages of task anticipation, this work also quantifies the level of cooperation and performance augmentation and shows some interesting patterns as a result of task anticipation. To conduct the experiments, a simulator was developed, changes were made to the Amaya Workflow definition tool and CEMT's Workflow Engine was redesigned.

Keywords: *workflow*, anticipation, flexibility, cooperation, simulation.

1 INTRODUÇÃO

A fim de tornar o trabalho controlável e encorajar a comunicação entre empregados, Sistemas Gerenciadores de *Workflow* (SGWf) foram criados. Estes se tornaram uma nova classe de sistemas de informação, possibilitando construir, de maneira avançada, uma ponte entre o trabalho das pessoas e os sistemas de informação (AALST; HEE, 2002).

O gerenciamento do *workflow* refere-se às atividades gerenciais manuais e automáticas, como a modelagem de processos de negócio, a alocação de tarefas aos papéis que são assumidos por agentes, o controle das rotas das tarefas, controle do andamento do processo e envio de alertas aos agentes no caso de eventos especiais e excepcionais no sistema (STEINMACHER *et al.*, 2004c).

Os SGWfs tradicionais são, na maioria dos casos, utilizados para gerenciar processos de negócio bem estruturados do ponto de vista transacional. Por este motivo estes sistemas seguem uma rigorosa metodologia de engenharia de processos de negócio e especificação formal que levam a processos muito bem definidos, rígidos e não flexíveis.

Segundo Aalst e Hee (2002) esta inflexibilidade e rigidez dos produtos atuais acabam por espantar muitos usuários potenciais. Hagen e Alonso (1999) apresentam o mesmo ponto de vista, dizendo que inflexibilidade e rigidez acabam por tornar as tecnologias de *workflow* atuais restritas a apenas alguns domínios de aplicação. Várias outras pesquisas apontam estes pontos como uma das grandes razões da resistência à utilização de SGWfs em alguns domínios de aplicação (ELLIS; KEDDARA; ROZENBERG, 1995; REICHERT; DADAM, 1998; JOERIS; HERZOG, 1999; MANGAN; SADIQ, 2002; REICHERT; DADAM; BAUER, 2003).

Visto isto, fica evidente a necessidade de fornecer meios para relaxar algumas das restrições impostas pelos SGWfs tradicionais. Uma destas restrições é a impossibilidade das tarefas produzirem, disponibilizarem ou utilizarem resultados intermediários, sendo as tarefas vistas como caixas pretas. Relaxando esta restrição, as tarefas poderiam ser disparadas antes mesmo da conclusão de suas antecessoras, ocorrendo a chamada antecipação de tarefas (GRIGORI, 2001). Este é um dos objetivos deste trabalho, encontrar maneiras de flexibilizar o modelo de *workflow* tradicional, de forma a possibilitar que tarefas seqüenciais possam trocar resultados intermediários, iniciando sua execução de maneira antecipada.

A antecipação implica no aumento da cooperação entre tarefas e em melhor desempenho com relação ao tempo de execução dos processos. O aumento na cooperação está relacionado diretamente à troca de resultados, permitindo que duas ou mais tarefas trabalhem sobre instâncias de um determinado dado que ainda está sendo produzido. O melhor desempenho deve-se ao paralelismo (parcial) criado

entre tarefas que deveriam executar de maneira estritamente seqüencial, devido ao disparo das tarefas antes do previsto.

O aumento na cooperação citado no parágrafo anterior pode influenciar negativamente o desempenho do processo, causando aumento também no esforço de comunicação exigido. Isto pode tornar o uso da antecipação uma estratégia não muito útil às organizações. Para tentar minimizar este tipo de problema podem ser utilizados elementos de percepção (*awareness*) que suportem tal antecipação, de maneira a controlar o esforço inserido pelo aumento da cooperação, sem comprometimento do desempenho.

Com base nisto, este trabalho tem como objetivo apresentar um estudo sobre os meios de flexibilizar-se os SGWfs de modo a prover a antecipação de tarefas, tendo como foco principal verificar o desempenho de sua utilização em processos cooperativos.

O estudo resultou na identificação e classificação das estratégias de gerenciamento de antecipação de tarefas. Esta classificação foi feita de acordo com a fase do ciclo de vida em que a flexibilização é aplicada (tempo de definição de processo / tempo de execução do processo).

Duas estratégias são especificadas: *a priori* e *a posteriori*. A primeira delas é totalmente vinculada à definição dos processos e deve ser prevista no modelo. Esta forma de antecipação pode ser utilizada em qualquer SGWf existente, porém, não oferece flexibilidade pré-determinada, continuando este estático e exigindo grande esforço de modelagem. A outra maneira de prover antecipação está relacionada à execução dos processos, deixando a decisão por antecipar ou não antecipar para a fase de execução dos processos. Esta abordagem é mais complexa e exige alterações na máquina de execução do SGWf, porém, oferece grande flexibilidade ao processo e não exige esforço extra durante a modelagem.

Após a especificação das estratégias de gerenciamento, foi implementado um simulador capaz de prever a antecipação de tarefas. Este foi utilizado para a condução de simulações que mostram o desempenho dos processos comparando a utilização de antecipação e a execução tradicional. Estas simulações apresentam os resultados com relação ao tempo de execução dos processos. Elas levam em consideração tanto a antecipação como possíveis perdas no desempenho devido ao esforço adicional inserido pela comunicação e pela troca de resultados intermediários.

Por fim, são propostas alterações junto ao *Amaya Workflow* (TELECKEN, 2004) para permitir a definição de processos e na máquina de *workflow* implementada por França (2004) para que estes suportem a antecipação de tarefas.

1.1 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o capítulo 2 discute os conceitos básicos relacionados ao trabalho. O capítulo 3 traz uma revisão bibliográfica dos trabalhos relacionados à flexibilização de SGWfs. O capítulo 4 apresenta a antecipação de tarefas e as possíveis maneiras de aplicá-la, além de uma discussão com relação aos elementos de percepção que dariam suporte à antecipação. O capítulo 5 apresenta as simulações realizadas, bem como seus resultados. No capítulo 6 são apresentadas as implementações realizadas. E, finalmente, no capítulo 7 são apresentados as conclusões e trabalhos futuros.

2 CONCEITOS E TECNOLOGIAS UTILIZADAS

Este capítulo apresenta os conceitos e tecnologias que são utilizadas no decorrer do presente trabalho. Tendo em vista os objetivos deste trabalho, serão apresentados a tecnologia e os conceitos inerentes a *workflow* e sistemas gerenciadores de *workflow*, seus elementos, nomenclatura e funcionalidades. Será também apresentada uma seção relacionada à percepção (*awareness*) e suas possíveis classificações.

2.1 *Workflow*

Esta seção traz definições básicas relacionadas a *workflow* que servirão como referência para o restante do trabalho. Pretende-se adotar, sempre que possível, as definições sugeridas pela *WfMC* (2004), sendo extraídas basicamente do glossário (WFMC, 1999a) e do manual de referência (WFMC, 2004).

2.1.1 *Workflow* - Conceito

Desejando encontrar uma definição para o termo *workflow*, é possível verificar que a literatura apresenta um grande número de variações. Aalst e Hee (2002) definem *workflow* como sinônimo de processo de negócio, sendo uma rede de tarefas com regras que determinam uma ordem parcial através da qual as tarefas devem ser executadas.

Já a *Workflow Management Coalition* (WFMC, 2004) define o termo como a automação de um processo de negócios, por inteiro ou em parte, durante o qual documentos, informações e tarefas são passadas de um participante para outro se respeitando um conjunto de regras procedimentais.

Casati *et al.* (1996) dizem que *workflow* pode ser definido como um conjunto coordenado de tarefas (seqüenciais ou paralelas) que são interligadas com o objetivo de alcançar uma meta comum.

Para Grigori (2001), é a representação formal de um processo, que especifica principalmente as tarefas e suas interdependências, os recursos e os programas associados a cada uma das tarefas.

Apesar do grande número de diferentes definições, todas elas apresentam *workflow* como sendo algo que automatize a execução de tarefas. Nesta execução, encontram-se os outros elementos citados nas definições: interdependência entre tarefas, fluxo de informações, programas e recursos associados.

2.1.2 Conceitos Básicos

Nesta sub-seção serão apresentados os conceitos e a terminologia associados a *workflow* que serão utilizadas no presente trabalho. A Figura 2.1 mostra o

relacionamento entre os conceitos propostos pela *WfMC*, com algumas adaptações para as nomenclaturas utilizadas no decorrer do presente trabalho.

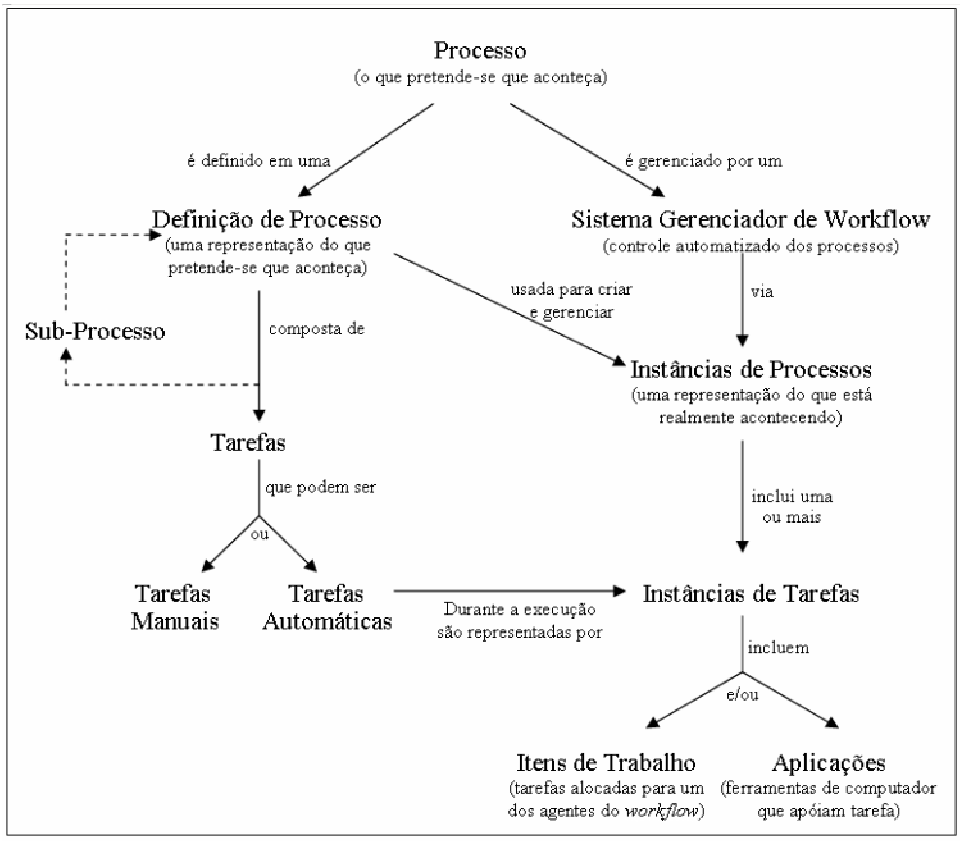

Figura 2.1: Relacionamento entre conceitos básicos de *workflow* (WFMC, 1999)

Processo (ou *processo de negócio*) nada mais é que um conjunto de um ou mais procedimentos ou tarefas que, coletivamente, têm um objetivo comum. Normalmente estão no contexto de uma estrutura organizacional definindo papéis e seus relacionamentos. Os processos podem variar significativamente em aspectos como tempo de duração e abrangência dentro das organizações. As tarefas englobadas nos processos podem variar de programas de computador até tarefas humanas (reuniões, tomadas de decisão, escrita de relatórios). Um *processo* é definido em uma *definição de processo*.

Uma *definição de processo* é a representação de um processo de maneira que este possa ser manipulado automaticamente. Ela consiste em uma rede de *tarefas* orientadas. Esta definição deve conter critérios para indicar o início e o fim do processo, assim como informações sobre cada tarefa, como agentes participantes, fluxo de dados e condições de disparo das tarefas (fluxo de controle), de maneira que seja possível atingir o objetivo do processo.

Cada *processo* pode ser composto, de maneira hierárquica, por vários *sub-processos*. Estes *sub-processos* são processos que são disparados de dentro de outros, fazendo parte deste. Este conceito é de grande utilidade dentro de um *workflow*, pois permite a reutilização e a modularização de processos. Cada um dos *sub-processos* contém sua própria definição de processos.

Um *processo* pode gerar várias *instâncias de processo*. Cada uma destas *instâncias* tem suas próprias características, que podem (ou não) ser diferentes das outras *instâncias* de um mesmo processo. Por exemplo, um processo pode ser aplicado em uma empresa de confecção, onde se produzem calças, saias e camisetas. Para cada um destes artefatos a serem produzidos é criada uma *instância* diferente, com suas próprias características. Cada instância de processo exibe um estado interno que representa a evolução do processo e suas tarefas.

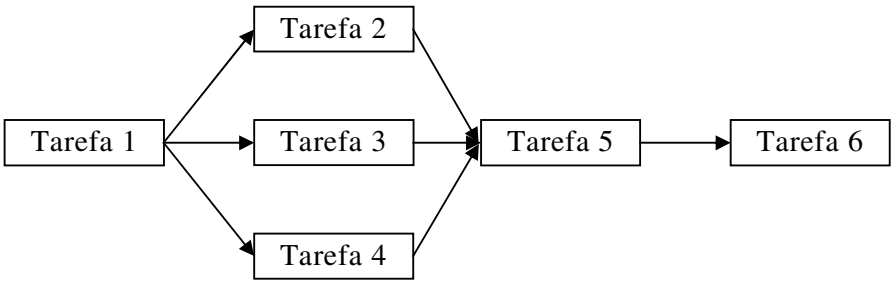

Figura 2.2: Exemplo básico de definição de processo

A *WfMC* define uma *tarefa* como a descrição de uma parte do trabalho que forma um passo lógico dentro de um processo, podendo ser manual ou automática. Uma tarefa é geralmente a menor unidade de trabalho considerada em um processo durante sua execução. Porém, esta pode resultar em vários itens de trabalho alocados para um agente. Aalst e Hee (2002) corroboram com a definição estabelecida pela *WfMC,* dizendo que uma tarefa é a menor unidade lógica de trabalho, sendo esta indivisível (atômica).

Neste trabalho, será utilizada a definição de que uma tarefa é a menor unidade de trabalho em uma definição *de processo,* porém, não serão consideradas unidades de trabalho *atômicas.* É justamente neste ponto que está o objetivo do presente trabalho, na possibilidade de tornar as tarefas unidades de trabalho divisíveis, quebrando a atomicidade das mesmas e tornando possível a antecipação de tarefas.

Uma tarefa requer recursos humanos (manuais) ou máquinas (automáticas) para apoiar sua execução. Quando humanos são necessários, a tarefa deve ser alocada para um agente que participe do processo. Aalst e Hee (2002) incluem ainda mais um tipo de tarefa: as *semi-automáticas*. Estas seriam executadas por um recurso humano apoiado por uma aplicação, não podendo ser classificada nem como manual, nem como automática.

A definição de uma tarefa inclui ainda a especificação dos dados de entrada e de saída, a aplicação que deve ser utilizada para realização da tarefa, quais as qualificações necessárias para que um agente possa executá-la e o meta-modelo para determinar se a tarefa está concluída ou não.

Uma *instância de processo* é composta por *instâncias de tarefas* que compõem tal processo. Cada instância de tarefa representa uma invocação única de uma tarefa, relativa à exatamente uma instância de processo e utiliza os dados

associados àquela instância de processo. Muitas instâncias de tarefa podem ser associadas a uma instância de processo, mas uma instância de tarefa não pode ser associada com mais de uma instância de processo.

2.1.3 Fluxo de Controle

Como dito anteriormente, um *workflow* descreve um processo, suas unidades de trabalho (tarefas) e também a maneira com que estas tarefas estão ligadas para chegar a um objetivo comum. É justamente esta maneira com que as tarefas estão ligadas que definem o fluxo de controle de um *workflow*.

Quando da definição do processo são definidas todas as tarefas que podem potencialmente suceder uma outra. Isto é, fica definida a seqüência em que as tarefas devem ser executadas e todos os possíveis caminhos que uma instância de processo pode seguir.

Os sucessores de uma determinada tarefa em uma instância de um processo vão depender das regras que controlam cada transição existente entre as tarefas. Estas regras podem ser chamadas de condições de transição e podem referir-se a dados relevantes do próprio *workflow*, aos dados de entrada da tarefa, a variáveis do sistema (como data e hora atuais) e também atender a eventos externos. A *WfMC* chama estas regras de pré-condições. Então, a cada conector existente entre tarefas, existe uma condição de disparo associada.

Existem quatro situações que devem ser consideradas quando se fala em fluxo de controle em um *workflow* (AALST; HEE, 2002):

- **Fluxo Seqüencial:** ocorre quando tarefas são executadas uma após a outra, isto é, seqüencialmente. Existe uma dependência clara entre tarefas que obedecem este tipo de fluxo. A Figura 2.3 exemplifica este tipo de fluxo.
- **Fluxo Paralelo:** ocorre quando duas ou mais tarefas precisam ser executadas simultaneamente ou em qualquer ordem. Neste caso as tarefas são executadas de forma que uma não influencie a outra. Este comportamento é obtido através do uso de um elemento do tipo *AND-split* para disparar as tarefas em paralelo, e um elemento do tipo *AND-join* para sincronizá-las. Tal fluxo é apresentado na Figura 2.1.
- **Fluxo Condicional:** quando apenas uma dentre duas ou mais tarefas deve ser escolhida para ser disparada, é utilizado este tipo de roteamento. A escolha da tarefa a ser disparada depende das propriedades de cada instância de processo. Este comportamento é conseguido através do uso dos elementos *OR-split* e *OR-join*. A escolha a ser realizada restringe-se a uma das tarefas que segue o *OR-split*. O fluxo condicional é mostrado na Figura 2.5.
- **Fluxo Iterativo:** em uma situação ideal, uma instância de tarefa não deve ser executada mais de uma vez por instância de processo. Porém, em certas ocasiões é necessário que haja um ciclo, repetindo a execução de uma ou mais tarefas, até que uma determinada condição seja satisfeita. Este tipo de fluxo pode ser entendido como um tipo especial de fluxo condicional. A Figura 2.6 exemplifica o fluxo iterativo.

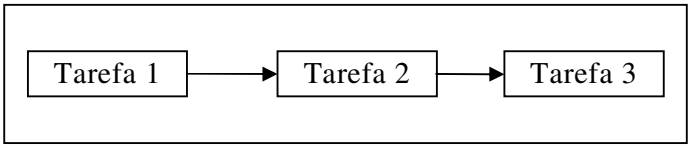

Figura 2.3: Exemplo de tarefas seguindo um fluxo seqüencial

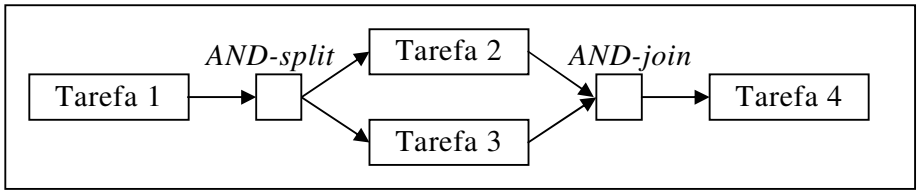

Figura 2.4: Tarefas obedecendo a um fluxo paralelo

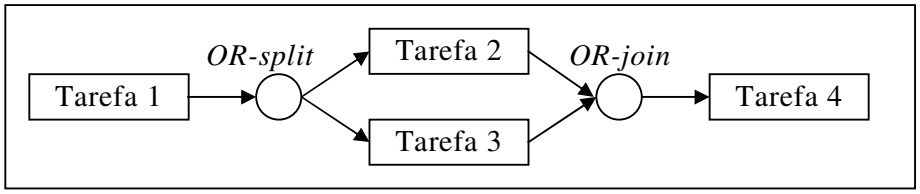

Figura 2.5: Utilização de elementos *OR-join* e *OR-split* provendo condicionalidade

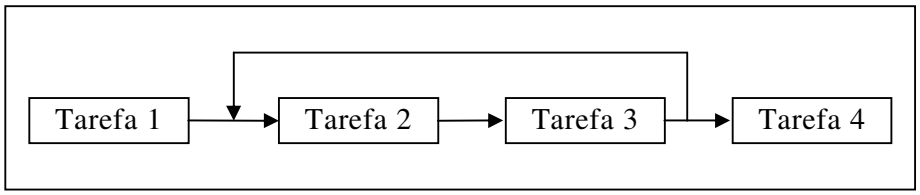

Figura 2.6: Fluxo iterativo, que pode ser definido através da condição de saída da tarefa

2.1.4 Fluxo de Dados

A definição do fluxo que os dados deve percorrer dentro de um processo deve ser realizada durante a modelagem das tarefas. O fluxo é estabelecido de acordo com as definições dos dados de entrada e de saída de cada uma das tarefas do processo.

Um conector de dados é "criado" entre duas tarefas quando o dado de saída de uma delas é um dado de entrada da outra. Isto é, se uma Tarefa A produz um determinado dado *x* que servirá de entrada para uma Tarefa B, é criado um conector de dados entre essas duas tarefas. Este conector de dados tem como função representar a dependência com relação ao dado *x* existente entre as tarefas A e B.

Na Figura 2.7 pode-se verificar tanto o fluxo de dados como o fluxo de controle de um processo. No caso, as linhas tracejadas representam o fluxo de dados e as linhas cheias representam o fluxo de controle.

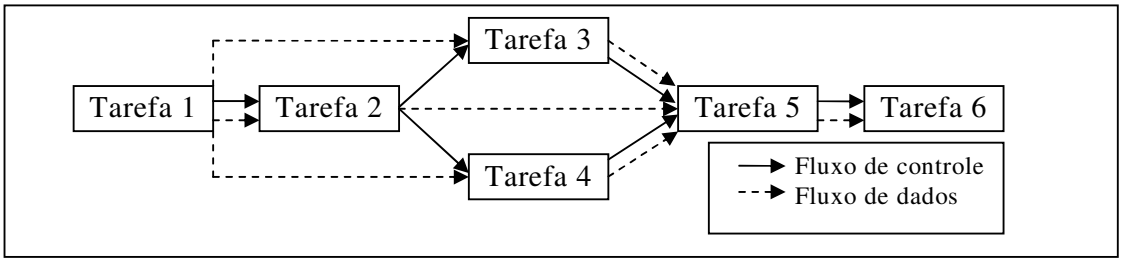

Figura 2.7: Processo explicitando conectores do fluxo de dados e do fluxo de controle

Como se pode perceber, os conectores referentes ao fluxo de dados podem ou não ter relação com o fluxo de controle. Algumas características destes conectores são apresentadas abaixo:

- Entre duas tarefas pode haver quantos conectores de dados quantos forem precisos.
- Uma tarefa pode ter quantos conectores de saída e de entrada quantos forem necessários;
- O sentido dos conectores deve seguir sentido do fluxo das tarefas (fluxo de controle). Isto é, uma tarefa deve sempre enviar dados para uma sucessora (direta ou indireta);
- Uma tarefa pode ter conectores de saída representando o envio de um mesmo elemento de dados para tarefas diferentes;
- Uma tarefa pode ter conectores de saída representando o envio de elementos de dados diferentes para uma mesma tarefa.

2.1.5 Arquitetura de um Sistema Gerenciador de *Workflow*

Segundo a WfMC (WFMC, 2004; WFMC, 1999) a estrutura de um projeto de automação de *workflow* deve seguir, basicamente, a arquitetura mostrada na Figura 2.8.

De maneira simples, esta arquitetura determina a existência de dois tempos funcionais: o tempo de construção e o tempo de execução de um *workflow* (TELECKEN, 2004). No tempo de construção, o modelo do processo de *workflow* é criado, e no tempo de execução, o processo de *workflow* é executado.

Através de uma ferramenta de definição, os designers de *workflow* podem gerar uma definição de processo. A definição de processo deve conter todas as informações que uma máquina de *workflow* necessita para gerenciar, controlar e executar um *workflow*. Após ser gerada, a definição de processos pode ser enviada para uma máquina de *workflow*. Primeiramente, a máquina de *workflow* irá interpretar a definição de processos e, em seguida irá controlar, gerenciar e executar o *workflow* descrito na definição de processos.

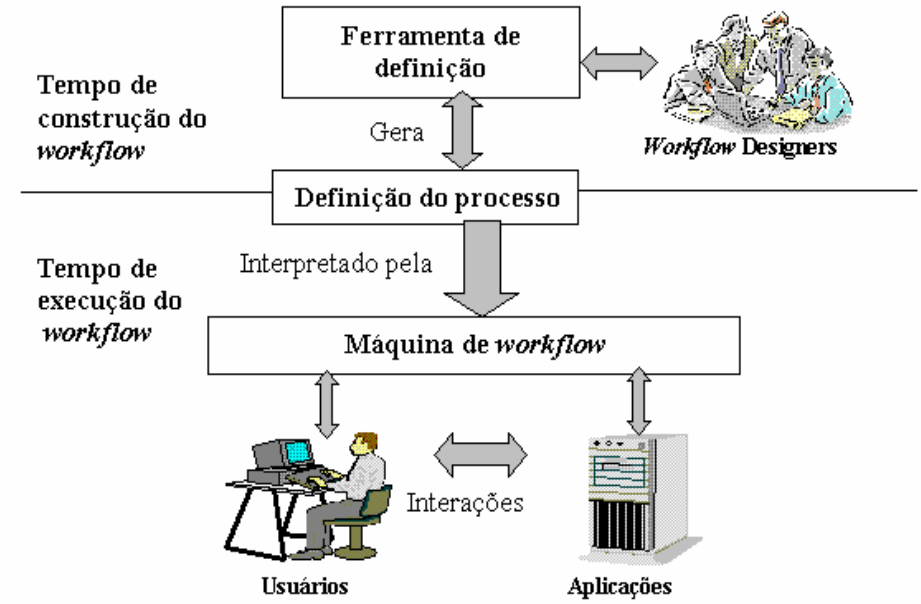

Figura 2.8: Estrutura Básica de um *workflow* proposta pela WfMC (Telecken, 2004)

A arquitetura da Figura 2.9 mostra que a definição de processo pode referenciar aplicações e estruturas da organização. Após ser modelada, a descrição do processo é interpretada por uma máquina de *workflow*. Este software é responsável pelo controle, gerenciamento e execução do *workflow* modelado. A máquina de *workflow* pode:

- utilizar/referenciar dados da organização;
- invocar aplicações externas;
- manter os dados de controle do *workflow* (são os dados necessários para controlar e manter o funcionamento do *workflow*);
- usar os dados relevantes do *workflow* (são os dados obtidos pela máquina de *workflow* através de recursos externos); e
- manter uma lista de trabalho (uma lista de tarefas que é associada a cada participante do *workflow*).

O usuário interage com o *workflow* através de sua interface. Ele vê em sua lista de trabalho as tarefas que podem ser disparadas e as executa invocando ou utilizando aplicações externas sempre que for necessário. Entre o usuário e a lista de trabalho há um software de intermediação chamado de manipulador da lista de trabalho.

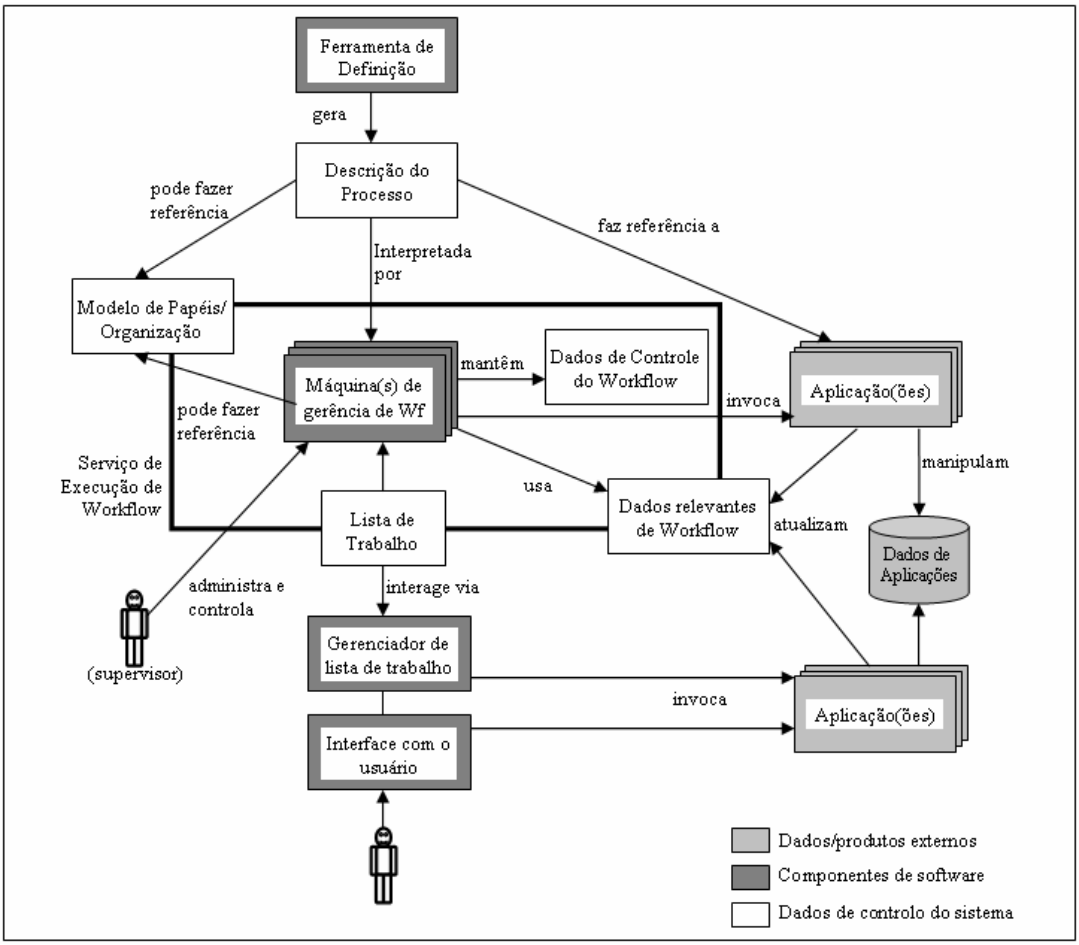

Figura 2.9: Arquitetura detalhada de um sistema de *workflow* (WFMC, 1995)

As aplicações externas podem ter suas próprias bases de dados e interagem com o *workflow* atualizando os dados relevantes e executando tarefas ao serem invocadas pelos usuários, pela máquina de *workflow* ou pelo manipulador da lista de trabalho. São exemplos de aplicações externas: editores de texto, servidores de e-mail, editores gráficos, etc.

2.1.6 Modelo de Referência para SGWfs

A arquitetura genérica de aplicações de *workflow* apresentada na seção anterior serviu de base a construção do modelo de referência para SGWfs (WFMC, 2004). Isso foi feito através da análise de elementos que compõem a estrutura genérica dos SGWfs e da análise das possíveis interfaces existentes entre eles. Além da identificação de cada interface, identificou-se também o comportamento de cada uma delas.

Como resultado destas análises, surgiu um conjunto de interfaces e formatos de intercâmbio de dados. A partir desse modelo, diferentes produtos de *workflow* podem alcançar variados níveis de interoperabilidade e comunicação que permitem a coexistência e a interação de diversos produtos de *workflow* com o ambiente usuário, como mostrado na Figura 2.10.

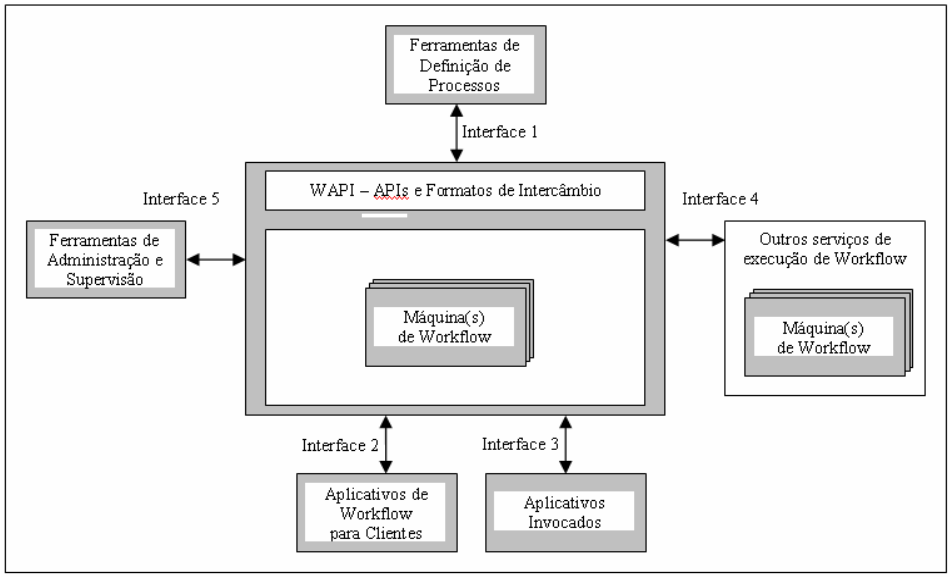

Figura 2.10: Modelo de Referência WfMC (WFMC, 2004)

As interfaces com o serviço de execução de *workflow* são chamadas *Workflow Application Programming Interface* (WAPI) e os formatos de intercâmbio definem como o serviço de execução comunica-se com os outros componentes do sistema. As seções seguintes trazem uma discussão mais completa sobre cada uma das interfaces e componentes do modelo de referência.

2.1.7 XPDL

XML Process Definition Language (XPDL) (WFMC, 1999) é uma linguagem baseada em XML, especificada por um *XML-schema* proposta pela WfMC para descrever e intercambiar definições de processos de *workflow* entre diferentes produtos de *workflow*.

Este meta-modelo identifica as entidades de uso comum nos esquemas de *workflow* em geral e, através da transformação dos esquemas proprietários de cada produto para o meta-modelo proposto – utilizando uma folha de estilos XSL (W3C, 2004c), por exemplo – torna-se possível a comunicação entre sistemas de *workflow* distintos. Baseado num meta-modelo mínimo que descreve as entidades básicas de um processo, o XPDL consegue representar as características fundamentais dos atuais sistemas de *workflow*. Esta linguagem é o padrão de comunicação definido para a interface 1 prevista na Figura 2.10.

Um conceito de grande importância encontrado no modelo XPDL é a "Extensão". Ele é representado no modelo pelo elemento `<ExtendedAttribute>`. Através do "*ExtendedAttribute*" pode-se representar em XPDL informações não previstas no modelo. Isto torna possível abranger uma gama maior de ferramentas com funcionalidades distintas.

Quando o "*ExtendedAttribute*" é utilizado no mapeamento de um elemento não previsto para o formato XPDL, é possível obter-se como resultado uma representação em XPDL válida. Porém, outros sistemas que foram desenvolvidos para compreender uma definição de processo escrita em XPDL poderão não interpretar corretamente o "*ExtendedAttribute*" em questão, caso este não esteja

previsto. Caso não estejam previstos, os "*ExtendedAttributes*" são ignorados pelo sistema. Este elemento aumenta a flexibilidade quando trata-se da exportação de um processo para XPDL, porém seu uso no mapeamento de elementos diminui a interoperabilidade do sistema.

2.2 Mecanismos de Percepção

Percepção (*awareness*) é o conhecimento sobre o grupo e suas tarefas passadas e presentes, e constitui ponto vital para o trabalho cooperativo, sem o qual este trabalho fica descoordenado e perde em qualidade e eficiência (KIRSCH-PINHEIRO *et al.*, 2001).

Neste trabalho pretende-se utilizar mecanismos de percepção a fim de melhorar a coordenação e a motivar a cooperação entre tarefas. Nesta seção será apresentada uma breve discussão sobre as razões de utilizar mecanismos de percepção em ambientes em que se necessita coordenação e cooperação. Segundo Kirsch-Pinheiro *et al.* (2001) sem percepção, o trabalho cooperativo coordenado é quase impossível.

Fuks, Gerosa e Pimentel (2003) afirmam que a percepção, que é inerente ao ser humano, torna-se central para a comunicação, coordenação e cooperação de um grupo de trabalho. Os indivíduos tomam ciência das mudanças causadas no ambiente pelas ações dos participantes, e redirecionam as suas ações. Para a coordenação do grupo são essenciais informações de percepção. É importante que cada um conheça o progresso do trabalho dos companheiros: o que foi feito, como foi feito, o que falta para o término, quais são os resultados preliminares, etc. Então, a percepção ajuda a levar aos participantes as mudanças que ocorrem no processo, dando a eles a oportunidade de acompanhar todo o progresso do trabalho em que estão envolvidos.

A atuação coordenada e cooperativa dos agentes humanos torna-se uma tarefa trivial com o uso de elementos de percepção. Isto porque estes elementos possibilitam a compreensão do trabalho no contexto geral dos processos em que um agente está trabalhando. Estes elementos mostram ainda as oportunidades de comunicação e de troca de informações (e dados), apoiando o trabalho em grupo.

Um projeto adequado dos elementos de percepção possibilita que os agentes participantes tenham disponíveis informações necessárias para prosseguir seu trabalho, sem ter que interromper seus colegas para solicitá-las. Isto é de grande importância, pois os agentes não necessitam mudar de foco a cada vez que se faz necessária alguma solicitação ou troca de informações.

É esperado que, eliminando a troca de mensagens desnecessárias, seja possível reduzir o esforço de comunicação entre os participantes durante um processo cooperativo. Pode-se reduzir tais mensagens através do uso de mecanismos de coordenação especialmente desenvolvidos para este fim.

A utilização de um sistema de *workflow* para coordenação é ideal, pois permite deduzir alguns elementos de percepção sem a necessidade de intervenção humana. O processo de comunicação é enormemente simplificado, pois ele pode ser realizado em um passo, ao invés dos quatro necessários durante a comunicação usual entre dois agentes: (i) formulação da solicitação; (ii) envio da solicitação; (iii) processamento da solicitação pelo outro agente; e (iv) a edição da resposta por este último agente.

Em ambientes em que a cooperação é predominantemente assíncrona (caso dos SGWfs), o artefato compartilhado é essencialmente o único espaço compartilhado disponível aos participantes, sendo peça chave neste tipo de cooperação (KIRSCH-PINHEIRO *et al.*, 2001). Portanto, é extremamente importante deixar os agentes cientes das condições dos objetos que estão sendo (ou que serão) utilizados por eles durante a execução de suas tarefas.

Um ponto importante a ser tratado nesta seção diz respeito à quantidade de informação a ser provida pelo mecanismo de percepção. Uma interface mal projetada pode causar tanto a sobrecarga, quanto à omissão, por ser inadequada ao tipo de informação apresentada. Deve−se procurar por elementos de interface adequados. Estes elementos podem ser ícones ou cores associados a informações específicas, como papéis e participantes, ou gráficos representativos do progresso do trabalho, ou ainda uma combinação de elementos como estes. O importante é que estes elementos apresentem as informações de percepção de forma resumida, sem sobrecarga, nem perda de conteúdo significativo. Uma quantidade não gerenciável de informações dificulta a organização dos membros do grupo, ocasionando desentendimentos (FUSSEL *et al.*, 1998). Para evitar a sobrecarga, é necessário balancear a necessidade de fornecer informações com a de preservar a atenção sobre o trabalho. O fornecimento de informações na forma assíncrona, estruturada, filtrada, agrupada, resumida e personalizada facilita esta tarefa (FUKS; GEROSA; PIMENTEL, 2003).

2.2.1 Classificação dos tipos de elementos de percepção

Em (KIRSCH-PINHEIRO *et al.*, 2001) é apresentado um *framework* criado a partir da análise de vários mecanismos de suporte à percepção propostos na bibliografia. Este *framework* apresenta um conjunto de características importantes para o fornecimento de percepção dentro de ambientes síncronos e assíncronos. Estas características giram em torno de seis questões: o que, quando, onde, como, quem e quanto.

- **O que?** – Refere-se a quais informações devem ser fornecidas aos usuários. Estas informações estão ligadas principalmente a dois aspectos: tarefas e papéis. Estes aspectos constituem dois tipos de informações que devem ser característicos para o suporte à percepção. Dentro do contexto em que está inserido o usuário esta pergunta pode ser o que há para fazer, o que está atrasado, o que já foi concluído.

- **Quando?** – Percepção intimamente relacionada ao tempo, ordem cronológica, temporalidade. Refere-se a dois aspectos essenciais: quando ocorreram os eventos geradores da informação de percepção e quando se dá a apresentação ao usuário. As informações de percepção são geradas por eventos que ocorrem durante o trabalho do grupo, e de acordo com seu momento de ocorrência, estes eventos vão ser mais ou menos úteis à percepção. Pode−se dividir a ocorrência dos eventos em quatro momentos: o "passado", para eventos que ocorreram em um intervalo de tempo no passado; o "passado contínuo", para eventos que começaram no passado, mas que continuam válidos até agora; o "presente", para eventos que estão ocorrendo neste momento e o "futuro", representando as opções futuras para o grupo.

- **Onde?** – Esta questão deve ser feita quando o assunto é onde as informações são geradas e apresentadas, o espaço de trabalho e os objetos compartilhados, onde o trabalho deve ser executado. Desta maneira seria montado um ambiente virtual de trabalho, tornando possível a percepção do ambiente e do trabalho em grupo pelos integrantes como se estivessem realmente todos no mesmo ambiente físico.
- **Como?** – Refere-se à forma como a informação é apresentada ao usuário. De que maneira a interface deve interagir com o usuário. Os usuários não podem sentir-se soterrados pelas informações de percepção, nem tê-las omitidas. Estas informações devem ter natureza passiva, surgindo direto das tarefas de cada pessoa, ao invés de ser gerenciada ou procurada explicitamente.
- **Quem?** – Esta pergunta faz referência a quem está trabalhando no momento. Este é um conhecimento importante para o andamento das tarefas no grupo, pois age como facilitador da cooperação, estimulando a comunicação.
- **Quanto?** – Esta questão destaca uma característica essencial do suporte à percepção: quanta informação deve ser apresentada às pessoas para lhe prover percepção sobre o processo e o grupo. O ponto chave desta questão é fornecer a quantidade certa de informações sem causar nem a insuficiência, nem a sobrecarga.

Araújo (2000) apresenta uma classificação que tem como base os itens que um individuo necessita para ter uma noção completa da interação que participa. Sendo assim, é preciso que ele possa perceber o *contexto social* que perfaz os grupos dos quais é membro; o *contexto de tarefas* a serem realizadas e as alterações ocorridas no *espaço de trabalho* durante a interação. Isto tudo se levando em conta que todos os tipos de informação variam no tempo, tendo de ser apresentadas aos participantes acrescidas da dimensão de tempo.

A classificação de Araújo (2000) é detalhada a seguir:

- **Percepção Social**: Os usuários devem ser capazes de reconhecer o grupo no qual estão inseridos, obter informações sobre seus participantes e estabelecer conexões sociais com estes visando o bom andamento da interação. São enumerados alguns elementos de percepção social:
    - *Composição*: informações que dizem respeito justamente ao conhecimento dos grupos a que pertence, sabendo seu papel e sua responsabilidade junto a cada grupo.
    - *Localização*: informação sobre a localização geográfica dos participantes pode ser importante em alguns tipos de aplicações cooperativas. Esta informação pode compreender grandes ou pequenas distâncias.
    - *Presença*: cada usuário deve obter informações sobre quais membros do grupo estão interagindo com ele. O sistema deve ser capaz de oferecer informações sobre a ação dos demais membros do grupo sobre o espaço compartilhado.

- *Proximidade*: o objetivo dessa informação é determinar o grau de proximidade (física, entre papéis, entre responsabilidades) dos participantes de um determinado trabalho em grupo.
- *Disponibilidade*: em alguns casos faz-se necessário oferecer informações sobre quais usuários podem ser contatados pelos demais participantes em um determinado momento.
- *Emoção*: informações sobre a reação emocional dos participantes frente a um determinado fato.

- **Percepção de Tarefas**: Os usuários devem estar conscientes sobre os objetivos e a estrutura do trabalho a ser realizado. Alguns tipos de informação sobre as tarefas também são enumerados:
  - Estrutura: o conhecimento da estrutura do trabalho (ordem das tarefas) e a noção de responsabilidade de cada membro em relação às tarefas permitem aos participantes do trabalho direcionarem suas contribuições. Conhecer as dependências das tarefas é de grande importância para um bom andamento de um trabalho cooperativo.
  - Status: este tipo de elemento de percepção inclui informações sobre o andamento das tarefas: em execução, executada, aguardando liberação.

- **Percepção de Espaço de Trabalho**: Cada usuário deve estar consciente das ações realizadas pelos demais participantes no processo de trabalho. Com estas informações cada usuário pode idealizar suas contribuições junto ao espaço compartilhado. Também são enumeradas algumas informações inerentes a esta classe de percepção:
  - *Status dos objetos de trabalho:* os participantes devem estar cientes do estado dos objetos contidos no espaço compartilhado. Informações deste tipo são: histórico e informações sobre a conclusão de um objeto.
  - *Ações:* os usuários podem ter acesso a informações sobre as ações realizadas no espaço compartilhado. Quem as realizou, quando, porque e os possíveis impactos de sua realização.
  - *Posição:* os participantes de uma interação cooperativa precisam ter conhecimento do que os outros participantes estão fazendo no espaço compartilhado.

Cada ambiente pode oferecer apenas o subconjunto de informações que forem necessários à conclusão dos objetivos da aplicação que está sendo apoiada pelo mecanismo de percepção. Isto é, dependendo do tipo de cooperação que se deseja, pode-se omitir ou dar mais ênfase a um determinado elemento de percepção, tendo em vista a melhoria na realização das tarefas. Além do que, certas informações são

mais úteis em ambientes de síncronos menos em ambientes assíncronos (por exemplo, posição no espaço de trabalho), e vice-versa (por exemplo, status das tarefas).

Em (GODART *et al.*, 2004) também é apresentada uma classificação dos elementos de percepção, porém de maneira mais objetiva, focando apenas aquela proposta no trabalho. As classes são:

- **Percepção de estado** (*state awareness*), que é baseada no estado dos dados que são compartilhados durante a execução do processo. É utilizado para apresentar ao usuário o aparecimento de novas versões ou conflitos existentes em documentos que estão sendo utilizados por ele.
- **Percepção de disponibilidade** (*availability awareness*), é baseada no conhecimento da presença física e na disponibilidade atual dos membros do grupo.
- **Percepção de processo** (*process awareness*), baseado no conhecimento sobre as tarefas e seu lugar e importância dentro do processo como um todo.

2.3 Considerações Finais

No presente capítulo foi apresentada a fundamentação teórica necessária ao entendimento do presente trabalho. Foram explorados os conceitos e padrões relacionados a *workflow* e SGWf, bem como mecanismos de percepção. Espera-se que os conceitos aqui apresentados consigam atingir o objetivo de servir como base para a compreensão do trabalho.

3 FLEXIBLIDADE EM SISTEMAS DE *WORKFLOW*

A busca por aumento na flexibilidade em sistemas de *workflow* é um tema muito explorado em pesquisa na atualidade. Tais pesquisas têm por objetivo, basicamente, tornar sistemas gerenciadores de *workflow* úteis a todas as áreas em que estes podem ser aplicados. A flexibilidade buscada vem de maneira a acabar com a estrutura extremamente rígida encontrada nos sistemas de *workflow* tradicionais. De acordo com os objetivos da presente dissertação, foi realizada uma revisão bibliográfica relacionada à flexibilização em SGWfs, enfatizando a antecipação de tarefas.

Este capítulo vem justamente apresentar as pesquisas e abordagens relacionadas à flexibilidade encontradas na literatura. A partir da análise das propostas encontradas, foi possível criar uma classificação básica do tipo de flexibilidade que as abordagens podem oferecer: flexibilidade *a priori* e flexibilidade *a posteriori*.

Flexibilidade *a priori* é aquele tipo de flexibilidade oferecida durante a modelagem dos processos. Este tipo de flexibilidade está relacionada ao tempo de definição de processos, criando meios de tornar a execução de um processo menos restritiva através de novas maneiras de modelar processos.

Já nas abordagens onde é proposta a flexibilidade *a posteriori* são apresentadas maneiras de tornar os SGWfs menos rígidos através de mudanças em tempo de execução dos processos. Estas mudanças dizem respeito a alterações dinâmicas na definição dos processos e alterações no modo de execução dos processos.

Foi encontrada ainda uma abordagem que se apresenta mais abrangente, contemplando tanto elementos para prover flexibilidade *a priori* como para prover flexibilidade *a posteriori*.

3.1 Flexibilidade *a priori*

A primeira abordagem a ser apresentada é a proposta do operador *Coo* (GODART; PERRIN; SKAF, 1999), que foi desenvolvido a fim de tornar os modelos de *workflow* capazes de representar as interações entre tarefas cooperativas. Tal operador não cobre todos os aspectos da cooperação, sendo, principalmente, dedicado ao gerenciamento de cooperação entre tarefas. Um exemplo da utilização deste operador é a resolução de exercícios por um estudante e a correção dos mesmos por um professor em um ambiente de ensino a distância. Os resultados dos exercícios podem ser enviados ao professor a qualquer momento, a fim de obter-se correções e sugestões por parte do professor. Esta interação pode ocorrer quantas vezes for necessário. O professor não é obrigado a corrigir todos

os resultados intermediários que lhe são enviados, sendo obrigado apenas a corrigir a versão final dos exercícios.

Tal interação é apresentada na Figura 3.1(a) com os operadores tradicionais de *workflow*. Porém, tal modelagem ainda oferece uma rigidez desnecessária na troca de resultados entre alunos e professor, obrigando que o último corrija todos os resultados intermediários dos exercícios enviados pelos alunos. A Figura 3.1(b) apresenta a utilização do operador *Coo* para este exemplo.

Além deste exemplo, que representa uma interação do tipo desenvolvedor/inspetor, ainda é possível modelar-se relações do tipo produtor/consumidor e escrita cooperativa utilizando tal operador.

O uso deste operador não implica em nenhuma restrição de quais objetos serão compartilhados ou a maneira como serão compartilhados. Porém, dois parâmetros são definidos a fim de configurar algumas interações. O primeiro deles define o sentido em que tal troca de resultados pode ocorrer (→, ←, ←→). O segundo parâmetro define se todos os resultados intermediários devem ser consumidos (*all*) ou não (*atdemand*). Ficando o exemplo acima, modelado da seguinte maneira: (exercícios, →, *atdemand*), (correções, ←,*atdemand*).

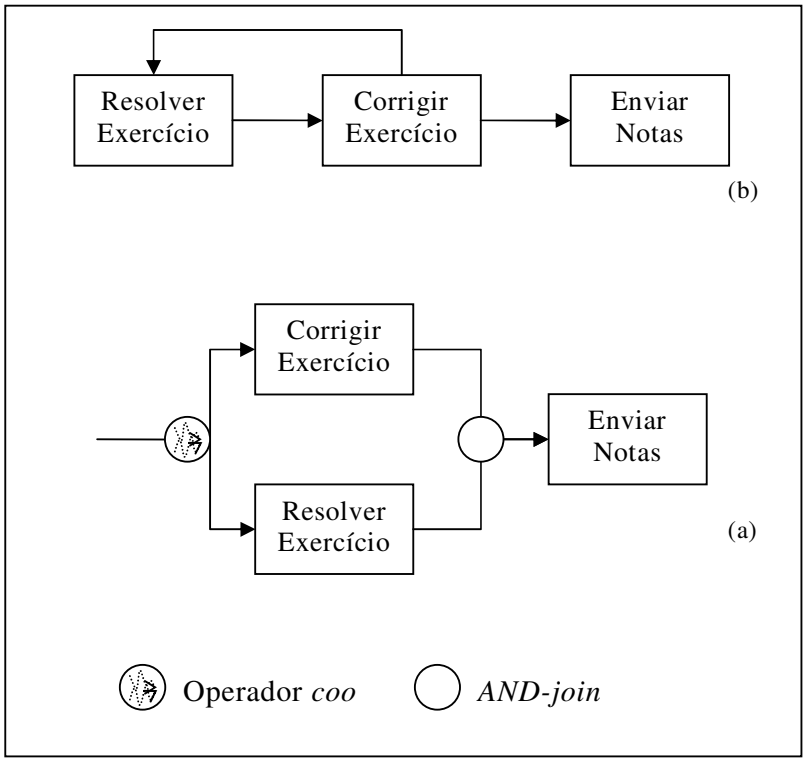

Figura 3.1: O uso do operador Coo (GODART; PERRIN; SKAF, 1999)

A criação de um operador para modelagem de tarefas que devem cooperar entre si é uma idéia muito interessante. Através deste operador pode-se mapear a interação entre tarefas, permitindo a troca de resultados e a antecipação de tarefas. Porém, a necessidade de criação de novos elementos de modelagem não é muito interessante. Além disso, ainda será necessária a criação de uma estrutura que seja

capaz de executar os processo de acordo com o comportamento especificado pelo elemento em questão.

A complexidade inserida à modelagem dos processos com as alterações propostas no *Coo* é grande, exigindo do responsável pela definição dos processos a configuração prévia dos meios com que se deve dar a troca de informações entre as tarefas.

Na área de modelagem de processos de software, foi encontrada a linguagem PROMENADE (RIBÓ; FRANC, 2002), que oferece um meio de formalizar os modelos de processos de software, visando aumentar a expressividade, padronização, flexibilidade e reuso dos mesmos. Esta linguagem apresenta meios para criação de dependências de fluxo de controle declarativas (a implementação de um componente *poderia* começar algum tempo depois do início da especificação e poderia terminar depois do final desta especificação) ao invés das imperativas (a implementação de um componente *deverá* começar assim que a especificação for concluída), baseado nos relacionamentos de precedência. Segundo os autores este tipo de relacionamento traria um maior grau de liberdade aos participantes do processo e um modelo menos prescritivo e mais flexível.

O PROMENADE pré-define quatro tipos de relacionamento de precedência básicos, definidos como um conjunto completo para a modelagem de processos de software básicos. São eles:

Tabela 3.1: Relacionamentos de precedência propostos no PROMENADE

Tipo	**Notação**	**Significado**
início	$s \rightarrow_{início} t$	*t* pode iniciar se, e somente se, *s* for iniciada antes.
fim	$s \rightarrow_{fim} t$	*t* pode ser concluída apenas se *s* for concluída anteriormente.
forte	$s \rightarrow_{forte} t$	*t* pode iniciar se, e somente se, *s* foi concluída com sucesso anteriormente.
feedback	$s \rightarrow_{fbk} t$	*t* pode ser re-executada após a conclusão sem sucesso de *s*.

Estes tipos de relacionamento são definidos de maneira declarativa, e podem ser combinadas a fim de criar-se relações de precedência derivadas. Isto permite que os usuários possam criar relacionamentos que melhor se encaixem às necessidades dos processos.

O objetivo da presente dissertação também está relacionada à flexibilização das condições de disparo do fluxo de controle. Porém no PROMENADE não existem meios para flexibilizar-se o fluxo de dados, por tratar de uma área afim e não especificamente de sistemas gerenciadores de *workflow*. Os novos tipos de relacionamento apresentados são interessantes, e são de simples utilização. Porém, também existe acréscimo de complexidade na modelagem com o uso destes novos relacionamentos.

Outra proposta de flexibilidade *a priori* é encontrada no *Freeflow* (DOURISH *et al.*, 1996). Esta abordagem também possibilita relacionamentos diferenciados entre tarefas. Porém os relacionamentos propostos no Freeflow são declarativos, capazes de separar as dependências entre tarefas (informações do sistema) e as informações do usuário. Isto é feito através de um modelo de estados para as tarefas que distingue entre estados do usuário e estados do sistema. Dentro deste sistema são definidos alguns tipos de relacionamentos como o "*after*" e o "*concurrent*", significando, respectivamente, que uma tarefa somente pode ser executada quando da conclusão de suas antecessoras (fluxo seqüencial tradicional) e que duas tarefas podem ser executadas concorrentemente (fluxo paralelo tradicional).

Nesta abordagem existe apenas a preocupação em criar novas dependências entre tarefas baseadas no fluxo de controle. Não existem meios para tratar o fluxo de dados nesta abordagem, não sendo possível prover troca de resultados intermediários. Além do que, criar relacionamentos declarativos insere muita complexidade à modelagem do processo, pois exige que os relacionamentos sejam criados antes de seu uso.

Joeris e Herzog (1998, 1999) defendem que a definição correta do comportamento das tarefas é a base para modelagem de *workflow* menos restritivos e para suporte a alterações dinâmicas. Neste sentido é apresentada uma maneira de especificar diferentes tipos de dependência no fluxo de controle, visando a definição de *workflow* menos restritivos. Tais tipos de dependência são definidos basicamente por regras do tipo E-C-A (Evento-Condição-Ação). O mecanismo de disparo trata as tarefas como se estas fossem componentes reativos, que encapsulam seu comportamento interno e interagem com outras tarefas através de passagem de mensagens/eventos.

As tarefas são definidas por atributo, parâmetro e restrições do processo, além da especificação de um comportamento externo independente de contexto e de um comportamento dependente de contexto. O comportamento independente de contexto (transacional ou não-transacional) é dado pela definição de um diagrama de estados (estados, decomposições e operações/transições que devem ser invocadas). Uma regra do tipo E-C-A pode ser definida para cada transição. Já o comportamento dependente de contexto é definido durante a definição do processo através de diferentes tipos de dependências do fluxo de controle e do agrupamento de tarefas. As dependências de fluxo de controle são refinadas de maneira a apoiar processos heterogêneos e flexíveis. Para tal são utilizadas regras E-C-A que estabelecem dependência entre tarefas (fim-início, início-início, deadline). Relações de grupo são utilizadas para aplicar um padrão de comportamento a um conjunto arbitrário de componentes.

Um exemplo de regra apresentado por Joeris (1999) é a dependência do tipo "*softsync*". Tal dependência permite omitir a execução de uma tarefa. Outro exemplo apresentado é a definição de um relacionamento de grupo do tipo "*exclusion*", que força os membros a executar as tarefas de maneira mutuamente exclusivas.

Processos cooperativos são suportados neste sistema através da possibilidade de integrar versões à semântica do modelo de *workflow* no nível conceitual. Versões de documentos podem ser liberadas para outras tarefas permitindo um fluxo de dados versionado e a troca de resultados intermediários. Além disso, o fluxo de

dados pode ser usado com a finalidade de fluxo de controle. A disponibilidade dos dados de entrada pode ser checada e as operações de liberação de fornecimento de saídas gerariam eventos, podendo outras tarefas reagir a estes eventos. A troca de resultados intermediários é realizada através do uso da dependência "*simultaneous*" que relaxa a condição de ativação da tarefa. Os resultados intermediários podem, então, ser passados de uma tarefa para outra sem que a primeira tenha sido concluída Os resultados intermediários podem ser fornecidos como versões, até que se conclua a tarefa.

A flexibilidade e a liberdade oferecida são os pontos fortes desta proposta, uma vez que é possível flexibilizar controle e dados através da formalização do comportamento do processo quando da definição do mesmo. Assim, é possível prever qualquer comportamento às tarefas. A dependência do tipo *simultaneous* prevê o comportamento necessário à antecipação de tarefas, permitindo que tarefas troquem resultados intermediários.

Porém, assim como nas propostas anteriores, o grande problema relacionado a esta abordagem está na necessidade de criação e utilização de novos relacionamentos quando da modelagem dos processos. Isso insere muita complexidade extra ao modelo.

Raposo *et al.* (2001) apresenta a proposta de flexibilizar os mecanismos de coordenação em tarefas colaborativas através da separação entre trabalho de articulação e o trabalho coordenado. Isto possibilitaria a alteração da política de coordenação, através da simples alteração dos mecanismos de coordenação, sem a necessidade de alterar o sistema de colaboração. Além disso, interdependências e seus mecanismos de coordenação podem ser reutilizados. É possível caracterizar diferentes tipos de interdependências e os mecanismos de coordenação para gerenciá-las. É então proposto um *framework* com interdependências genéricas possíveis de ocorrerem entre tarefas em processos colaborativos.

As interdependências propostas são divididas em: interdependências temporais (a seqüência com que as tarefas devem ser executadas para se chegar ao objetivo inerente ao processo) e interdependências de recursos (como o sistema lida com o acesso seqüencial ou simultâneo de múltiplos participantes a alguns objetos). O conjunto das interdependências temporais propostas no *framework* são baseadas nas relações temporais de Allen (1983). Ele define um conjunto de relações primitivas e mutuamente exclusivas que podem ser aplicadas sobre intervalos de tempo, tornando possível aplicá-las à coordenação de tarefas cooperativas (que geralmente são tarefas não instantâneas). A partir da adaptação das relações de Allen para a coordenação foram definidas as seguintes interdependências:

- T1 *equals* T2: Esta dependência estabelece que duas tarefas devem ocorrer simultaneamente, devendo iniciar e terminar juntas.
- T1 *starts* T2: Esta dependência pode ser interpretada de duas maneiras. A primeira maneira T1 e T2 devem ser disparadas juntas, e a T1 deve ser concluída antes de T2. A outra maneira relaxa a obrigação de T1 terminar antes.
- T1 *finishes* T2: Também existem duas possíveis interpretações para esta dependência. Uma delas diz que T1 e T2 devem terminar juntas e T1 deve ser disparada depois e T2. Numa segunda interpretação, a relação de início é relaxada, não importando a ordem de disparo.

- T1 *before* T2: Neste caso, existem três possibilidades:
    - T2 deve ser executada apenas uma vez a cada vez que T1 for concluída;
    - T2 pode ser executada várias vezes após a conclusão de T1.
    - T1 não pode mais ser executada após T2 iniciar sua execução, isto é, T1 deve ser executada obrigatoriamente antes de T2.
- T1 *meets* T2 : T2 deve ser iniciada imediatamente depois da conclusão de T1.
- T1 *overlaps* T2: Esta dependência também é dividida em duas outras. Na primeira T1 deve ser iniciada antes de T2 e concluída depois do término de T2. Na outra derivação a única restrição é que T1 seja iniciada antes do início de T2 e T2 seja iniciada antes da conclusão de T1.
- T1 *during* T2: estabelece que T1 deve ser executada totalmente durante a execução de T2.

As interdependências de recursos foram criadas de maneira independente das interdependências temporais, adicionando maior flexibilidade ao modelo. São definidas três tipos de interdependências básicas:

- *Sharing*: Um recurso deve ser compartilhado entre algumas tarefas. Representa a situação em que muitos agentes querem editar um mesmo documento. Esta dependência inclui conceitos de divisibilidade e concorrência.
- *Simultaneity*: Um recurso está disponível apenas se um certo número de tarefas o requisita simultaneamente. Esta dependência representa, por exemplo, uma máquina que somente pode ser usada com mais de um operador.
- *Volatility*: Indica se, depois do uso, o recurso estará disponível novamente. Por exemplo, uma impressora é um recurso não-volátil, enquanto uma folha de papel é volátil.

Raposo *et al.*(2001) traz uma gama de novas dependências com relação tanto ao fluxo de dados como ao fluxo de controle. É evidente que estas dependências são meios de prover a antecipação de tarefas, servindo de opções para o nível de flexibilidade que se deseja obter. Porém, muitas opções resultam no aumento da complexidade da modelagem dos processos, assim como nas outras abordagens já apresentadas.

3.2 Flexibilidade *a posteriori*

Uma das propostas de flexibilidade *a posteriori* é a abordagem de Nickerson (2003), que apresenta o problema que as transações longas podem causar. Para tal é proposto um modelo de *workflow* baseado em eventos, através do qual é possível alterar o andamento de uma tarefa longa. Isto se fez necessário, pois durante uma transação longa, o contexto em que a tarefa está envolvida pode mudar. O exemplo

dado no trabalho é da solicitação de um visto, que pode durar meses. Neste intervalo de tempo, o solicitante pode morrer. Se algo não for feito, a tarefa de avaliação do pedido de visto pode continuar, mesmo estando o solicitante morto. Porém, através de um evento externo, é possível interromper tal tarefa, economizando o tempo que ali seria dispensado.

A solução apresentada por Nickerson (2003) flexibiliza a restrição de isolamento das tarefas, através de eventos externos capazes de interromper uma tarefa. Na presente dissertação também são previstas alterações no modelo dos SGWfs para acabar com o isolamento das tarefas, porém para possibilitar de utilização de resultados intermediários fornecidos por tarefas antecessoras.

No sistema ADEPT (REICHERT; RINDERLE; DADAM, 2003a) é proposto um suporte a mudanças no modelo quando da instanciação dos processos. O ADEPT é um modelo conceitual de *workflow* baseado em grafos que possui uma base formal para sua sintaxe e semântica. Baseado neste modelo foi criado um conjunto completo e mínimo de operações de modificação que apóia os usuários na modificação da estrutura de instâncias de processos de *workflow* em execução, preservando sua corretude e consistência – $ADEPT_{flex}$ (REICHERT; DADAM, 1998). Tais verificações dizem respeito tanto ao fluxo de dados quando ao fluxo de controle.

O $ADEPT_{flex}$ compreende operações para:

- inserção de tarefas e blocos de tarefas;
- remoção de tarefas e blocos de tarefas;
- operações para avanço rápido do progresso do *workflow* (saltando algumas tarefas);
- serialização de tarefas previamente definidas como paralelas;
- paralelização de tarefas anteriormente seqüenciais;
- iteração e *rollback* dinâmico de uma região do *workflow*.

Na seqüência deste projeto, Reichert, Rinderle e Dadam (2003b) apresentam um *framework* para apoiar tanto a propagação de alterações dos modelos de *workflow* para as instâncias em execução deste *workflow* como de alterações *ad-hoc* de instâncias isoladas. Primeiramente são analisados os problemas que podem ser encontrados quando da alteração de um esquema e a respectiva propagação destas modificações às instâncias em execução. Dois problemas são identificados:

- Uma determinada alteração pode ser corretamente propagada para todas as instâncias correntes de um determinado esquema S, sem causar erros ou inconsistências?
- Assumindo que a primeira condição tenha sido satisfeita, que adaptações na instância tornam-se necessárias neste caso?

Para resolução de tais problemas são apresentados axiomas e teoremas que fundamentam a correta manipulação das modificações dinâmicas no *Workflow*.

Além de apresentar solução para a propagação de tais alterações e para mudanças *ad-hoc* em instâncias isoladas, conflitos entre instâncias e modificações no tipo são tratadas.

Alterações dinâmicas no modelo em tempo de instanciação e de execução são bem vindas, e podem ser de grande utilidade em processos com características intelectuais e cooperativas. Porém, tal abordagem esbarra na mesma rigidez encontrada nos sistemas tradicionais. É possível definir uma certa flexibilidade, porém, as tarefas continuarão sendo caixas pretas para os usuários, durante a execução. Os resultados somente poderão ser trocados ao final destas tarefas. Tornando as tarefas caixas brancas possibilitaria o disparo das mesmas antes do final de suas antecessoras, com resultados intermediários.

O projeto *Co-flow* (KRAMLER; RETSCHITZEGGER, 2000) traz uma abordagem de suporte a *workflow* inteligentes, construído sobre o TriGSflow. Nesta abordagem, agentes inteligentes são habilitados a adaptar e estender os modelos de processos pré planejados para suas situações específicas, sem influenciar os outros casos baseados no mesmo modelo. A adaptação dos modelos de processo podem ser iniciadas pelos agentes e as possibilidades de adaptação podem ser apoiadas por um sistema de recomendação, mostrando aos agentes as operações possíveis no contexto corrente.

Mais um trabalho que apresenta solução para o problema da alteração dinâmica de processos. É possível utilizar este tipo de abordagem como suporte à antecipação, porém, como dito anteriormente, isto demandaria muito custo e as tarefas continuariam como caixas pretas, executando em isolado.

Uma outra abordagem que oferece flexibilidade *a posteriori* é o *Coo-Flow* (GRIGORI; CHAROY; GODART, 2004), que apresenta uma tecnologia baseada em *workflow* para coordenar processos com interações criativas. Tal proposta baseia-se em análises realizadas nos domínios de arquitetura, engenharia, processos de software e aprendizagem cooperativa. Algumas conclusões relativas a estas análises são apresentadas:

(i) Processos criativos são descritos da mesma maneira que os processos de produção, porém sua interpretação depende da natureza da aplicação;

(ii) a interpretação do modelo depende fortemente da flexibilidade que se requer do fluxo de controle e do fluxo de dados; e

(iii) o gerenciamento de resultados intermediários é importante para suportar tal tipo de processos.

Inspirado nessas conclusões foi criado o *Coo-Flow*, que visa permitir justamente a antecipação de tarefas, provendo a flexibilidade necessária para a execução de processos cooperativos. Para possibilitar foi proposta primeiramente a criação de dois novos estados para as instâncias das tarefas.

Para permitir que as tarefas fossem disparadas antes da conclusão de suas antecessoras, passando assim para os estados propostos foi necessária a criação de novos estados para as tarefas e modificação do algoritmo de execução. Assim, foram propostas as seguintes estratégias de antecipação:

- *Antecipação livre:* as tarefas podem ser antecipadas a qualquer momento, tendo de respeitar apenas à dependência *fim-fim*, devendo aguardar a conclusão de suas antecessoras para serem concluídas;
- *Antecipação baseada no fluxo de controle:* as tarefas podem antecipar logo após suas antecessoras iniciarem sua execução, isto é, a dependência existente nesse caso é um *início-início*;
- *Antecipação baseada no fluxo de dados e no fluxo de controle:* as tarefas apenas podem iniciar sua antecipação quando suas antecessoras iniciarem a execução (dependência *início-início*) e seus dados de entrada (finais ou parciais) estiverem disponíveis.

A fim de oferecer a troca de resultados de maneira consistente, é utilizado o protocolo *Coo-protocol* (CANALS *et al*., 1998), especificado para as transações *Coo* (GODART; PERRIN; SKAF, 1999). Através deste protocolo são criados relacionamentos de dependência entre as tarefas que estão fornecendo e utilizando os resultados parciais. Basicamente, a função deste protocolo é não permitir que uma tarefa seja concluída utilizando um resultado parcial de um determinado dado, sem utilizar o resultado final deste dado.

Esta proposta serviu como motivação inicial desta dissertação, sendo a base de toda a pesquisa aqui realizada. Apresenta a possibilidade de levar à fase de execução dos processos a antecipação das tarefas. A proposta aqui apresentada utiliza alguns conceitos propostos no *Coo-flow*, e adiciona alguns outros. No *Coo-flow* não está previsto como será realizada a identificação de tarefas antecipáveis, e tampouco como os participantes envolvidos no processo serão estimulados ou informados da possibilidade de antecipar tais tarefas.

As alternativas de antecipação apresentadas no C*oo-flow* unificam a flexibilidade oferecida pelo fluxo de dados e de controle. As duas primeiras alternativas apresentadas não levam em consideração o fluxo de dados, permitindo o início das tarefas sem que haja dados de entrada disponíveis. Apenas a última (Antecipação baseada no fluxo de dados e no fluxo de controle) leva em conta os dados de entrada da tarefa. A criação de dependências de fluxo de dados e fluxo de controle independentes entre si poderia aumentar ainda mais as possibilidades de flexibilização dos sistemas gerenciadores de *workflow*. Este tipo de independência entre fluxo de controle e de dados é, de certa forma, proposta por Raposo *et al.*

Estes pontos serão abordados na presente dissertação, bem como testes e simulações que demonstram o ganho de eficiência na execução dos processos utilizando antecipação em relação à execução tradicional.

3.3 Uma abordagem mais abrangente

Além daqueles trabalhos que focam na flexibilidade *a priori* ou na flexibilidade *a posteriori*, foi encontrada uma abordagem que visa oferecer flexibilidade em todas as fases do ciclo de vida de um processo de *workflow*. Isto é, visa oferecer opções para flexibilizar os processos *a priori* e *a posteriori.*

Esta abordagem foi encontrada no projeto *Mobile* (HEINL *et al.*, 1999), que propõe uma divisão da flexibilidade provida pelos sistemas de *workflow* em flexibilidade por seleção e flexibilidade por adaptação, e apresenta como tais tipos de flexibilidades são implementadas dentro do projeto *Mobile*.

- *Flexibilidade por seleção* é provida pelo modelo de *workflow*, proporcionando diferentes interpretações válidas para um mesmo modelo de *workflow*, através das quais vários caminhos podem ser derivados. Este tipo de flexibilidade é útil quando um grande número de caminhos é antecipadamente conhecido. Existem dois métodos para implementar flexibilidade por seleção, diferenciados pelo momento em que são aplicados: modelagem avançada (antes da execução) e modelagem tardia (durante a execução). A flexibilidade por seleção é apresentada na Figura 3.2(a). O "início" e o "fim" representam o ponto de início e o ponto final da execução de um processo. O objetivo é ir do início ao fim. As linhas entre os dois pontos descrevem alguns possíveis caminhos de execução, especificados diretamente na definição do processo.

- *Flexibilidade por adaptação* decorre da necessidade de incorporar-se alguma modificação em um *workflow*. Para suportar tal flexibilidade é necessário que o SGWf possua ferramentas e funcionalidades que permitam alterar e integrar tais alterações ao modelo em tempo de execução. Novamente dois métodos foram identificados para tal flexibilidade: adaptação do tipo (*type adaption*) e adaptação da instância (*instance adaption*). A adaptação do tipo significa que todas as instâncias correntes não serão afetadas. Quando da adaptação da instância as instâncias correntes são afetadas. Neste caso é necessário decidir a quais instâncias as modificações serão aplicadas. No contexto da adaptação de instância, são utilizados os conceitos de versões e variantes. A Figura 3.2(b) demonstra como um conjunto de caminhos possíveis é estendido com um novo caminho através de adaptação do modelo de processo. A linha em negrito representa o novo caminho inserido por adaptação.

No contexto do *Mobile* a *flexibilidade por seleção* é implementada através da utilização de modelagem descritiva. A idéia básica de tal modelagem é utilizar um modelo de *workflow* orientado a perspectivas, onde a definição de cada perspectiva é totalmente independente das outras. As principais perspectivas utilizadas são a funcional (decomposição do *workflow* – o que deve ser feito), comportamental (fluxo de controle), organizacional (por quais agentes), informacional (fluxo de dados) e operacional (utilizando que aplicações). Cada uma dessas perspectivas é definida separadamente. Deixando de especificar algumas regras reduzirá as restrições quando da execução dos *workflow*, aumentando o número de caminhos (alternativas de execução) possíveis.

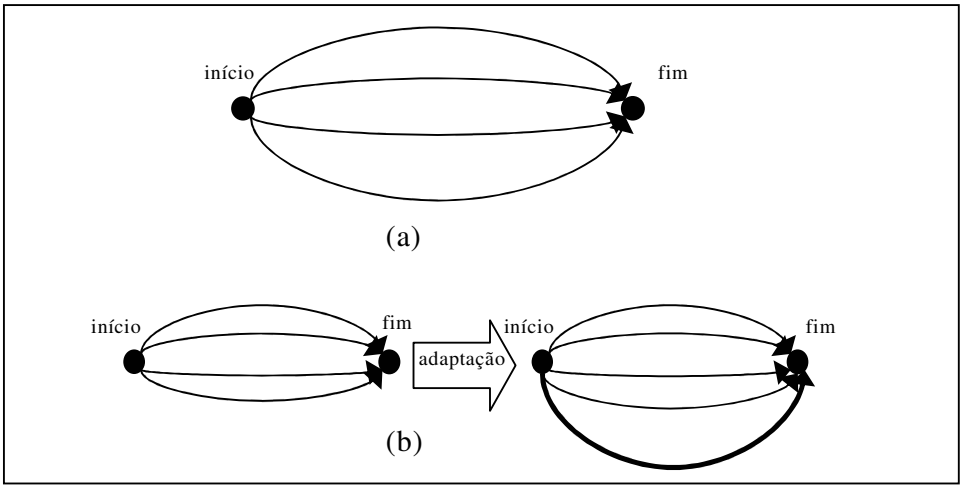

Figura 3.2: Inserindo (a) flexibilidade por seleção e (b) flexibilidade por adaptação (HEINL *et al.*, 1999)

Já a *flexibilidade por adaptação* é implementada pelo *Mobile* utilizando os conceitos de versão e variantes, seguindo um processo de adaptação composto de três passos:

1. Detecção das situações de erro durante a execução de uma instância;
2. Modificação do respectivo modelo de processo;
3. Adaptação de todas (ou um conjunto definido) das instâncias a serem modificadas.

O *Mobile* apresenta-se consistente e traz grande flexibilidade aos sistemas de *workflow* de maneira a permitir que as instâncias de um mesmo processo possam ser executadas de diferentes maneiras. Podendo isto ser previsto tanto pela definição independente de cada uma de suas perspectivas, quanto por alterações realizadas durante a execução de um processo, adaptando-o. Neste último caso o sistema é capaz de verificar a coerência do modelo alterado e verificar a aplicabilidade às instâncias em execução.

Apesar de poder ser uma alternativa para prover a antecipação de tarefas, o custo necessário para alterar o modelo durante a execução é muito grande. A complexidade para se conseguir verificar todas as instâncias em execução e realizar as alterações *on-the-fly* é altíssima.

3.4 Considerações Finais

Neste capítulo foi apresentada a revisão bibliográfica referente à flexibilidade em SGWfs. As abordagens estudadas foram classificadas naquelas que são previstas em tempo de definição dos processos (*a priori*) e aquelas que prevêem alterações durante a execução dos processos (*a posteriori*). De maneira geral, percebeu-se pelo grande número de pesquisas, que existe uma grande necessidade por tornar os SGWfs mais flexíveis, para permitir que *workflows* possam ser utilizados em mais domínios.

As abordagens com modificações em tempo de definição dos processos (*a* priori) estão focadas em sua maioria em alterações nas dependências intertarefas e na inserção de novos elementos de modelagem. Estas abordagens basicamente apresentam novos elementos ou formas de criar novos elementos de modelagem de processos de *workflow*. Estes elementos possibilitariam a definição de algumas características não previstas no modelo de definição de processos tradicionais. A antecipação de tarefas, inclusive, pode ser mapeada através do uso de algumas destas abordagens facilmente.

Aquelas abordagens que oferecem maneira de definir relacionamentos e comportamento de maneira descritiva (DOURISH *et al.*, 1996; JOERIS; HERZOG, 1998; 1999) aumentam muito as possibilidades de domínios a serem atendidas por sistemas de *workflow*, permitindo mapear qualquer comportamento desejado, inclusive a antecipação de tarefas. Algumas outras abordagens, basicamente criam novos elementos de modelagem e novas interdependências entre as tarefas. Apesar de apresentarem apenas alterações no modelo, é evidente que são necessárias modificações junto à máquina de execução para que esta reconheça estes novos elementos ou dependências. Portanto, além de inserirem complexidade à modelagem, estas novas abordagens necessitam SGWfs específicos para que se possa utilizá-los.

Já as abordagens *a posteriori* apresentam basicamente a facilidade de alterações dinâmicas dos processos. Isto permite que processos possam evoluir durante a execução, viabilizando a inserção, remoção e alteração das tarefas em instâncias de processos que já estão sendo rodadas pelo sistema. Certamente este tipo de abordagem pode ser utilizada para permitir a antecipação de tarefas, através da modificação automática da granularidade das tarefas de acordo com o tipo de resultado fornecido. Porém, a complexidade inerente à sua implantação é enorme, uma vez que são necessários mecanismos que garantam a consistência de todos os aspectos de um processo de *workflow* (dados, fluxo de controle, recursos).

A grande desvantagem da flexibilidade *a posteriori* é a complexidade de implementação das alterações da máquina de *workflow*. Porém, a flexibilidade oferecida por estas abordagens torna o sistema capaz de lidar com um maior número de situações não previstas em tempo de definição dos processos. Isso aumenta a quantidade de domínios que podem ser atendidos pelos sistemas.

Dentre as abordagens *a posteriori* é apresentada a abordagem de Grigori, Charoy e Godart (2004), que altera as condições de disparo e sugere novos estados para as tarefas, a fim de permitir a antecipação das mesmas. Esta abordagem será explorada no presente trabalho, e, como já dito, servirá de base para um dos mecanismos de antecipação apresentado no próximo capítulo.

4 ANTECIPAÇÃO DE TAREFAS

Segundo Grigori, Charoy e Godart (2004) antecipação é um conceito que, intuitivamente, significa que uma tarefa pode iniciar sua execução mesmo que antes do previsto. Mesmo sabendo que é possível antecipar o início de uma tarefa em decorrência da conclusão de suas antecessoras, antes do prazo previsto, não é esta a antecipação tratada neste trabalho. O conceito de antecipação aqui utilizado é iniciar uma tarefa mesmo que suas antecessoras diretas não tenham sido concluídas. Evidentemente, este início acontece durante a execução das tarefas antecessoras. Também, a conclusão das tarefas antecipadas depende da conclusão destas antecessoras.

A grande vantagem de utilizar este tipo de antecipação é o aumento na cooperação entre duas tarefas e a diminuição do tempo total de execução do processo. Isto se deve ao paralelismo (parcial) criado entre tarefas que deveriam executar de maneira estritamente seqüencial. Tal acréscimo no paralelismo é ilustrado na Figura 4.1. A Figura 4.1(a) representa o fluxo tradicional que o processo deveria seguir, condicionando o inicio de uma tarefa ao final de sua antecessora. A Figura 4.1(b) apresenta o fluxo do mesmo processo com a antecipação de uma tarefa (tarefa Revisão). Nesta última figura o gatilho da antecipação é o fornecimento de um resultado intermediário. A conclusão da tarefa Revisão tem como condição para conclusão o envio do resultado final da tarefa Escrita. No exemplo apresentado fica evidente o ganho com relação ao tempo de execução do processo, ao aumentar o paralelismo.

Em SGWfs tradicionais os processos de *workflow* são modelados de forma que as tarefas sejam caixas pretas e obedeçam a uma dependência do tipo *fim-início*. De certa forma, isto quer dizer que, nos SGWfs tradicionais, uma tarefa:

- é atômica;
- deve ser executada de maneira isolada;
- pode ser somente iniciada quando suas antecessoras forem concluídas; e
- precisa que seus dados de entrada (quando houverem) sejam resultados finais de suas tarefas antecessoras.

A fim de permitir a flexibilização de alguns destes pontos foi proposta a antecipação de tarefas (GRIGORI, 2001). Com isto as tarefas não mais são vistas como caixas pretas, havendo a possibilidade de troca de resultados.

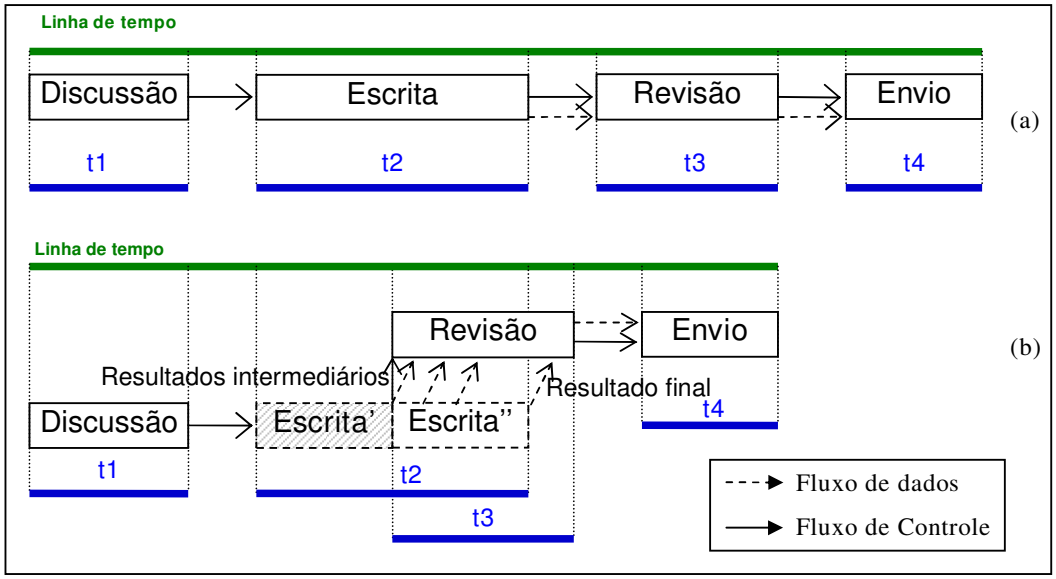

Figura 4.1: Exemplo básico do andamento de um processo utilizando antecipação de uma tarefa

Nesta dissertação foram estabelecidas duas possibilidades de gerenciar esta antecipação de tarefas: *a priori* e *a posteriori*.

A antecipação gerenciada *a priori* prevê alterações no processo em tempo de definição, sem a necessidade de qualquer alteração na máquina de *workflow*. Esta abordagem exige esforço extra durante a modelagem, mas pode ser utilizada em qualquer SGWf existente atualmente.

A antecipação gerenciada *a posteriori* prevê alterações na máquina de execução do SGWf, de maneira a acabar com as restrições de isolamento e atomicidade das tarefas. Para tal, será seguida a proposta de Grigori (2001) que define alterações no diagrama de estados das tarefas como a maneira de acabar com estas restrições. Para enriquecer o modelo, nesta dissertação é identificada a necessidade e é estabelecido um meio de identificar, em tempo de definição dos processos, as tarefas antecipáveis e o mapeamento desta identificação para a máquina de *workflow*. Também são apresentadas diferentes maneiras de lidar com dependências de controle e com o fluxo de dados e o mapeamento destas dependências para o diagrama de estados das instâncias de tarefas.

Nas abordagens apresentadas busca-se manter os padrões adotados pela WfMC. A abordagem *a priori* apresenta apenas uma nova "prática" de definição de processos, sem a inserção novos elementos de modelagem. Já as alterações apresentadas na abordagem *a posteriori* estão de acordo com aquelas previstas nas recomendações da WfMC.

4.1 Antecipação Gerenciada *a priori*

Como dito anteriormente a antecipação é um conceito intuitivo, assim como a possibilidade de prevê-la durante a modelagem do processo. No exemplo ilustrado na Figura 4.1 fica claro que é possível utilizar os SGWfs tradicionais para conseguir o ganho de paralelismo oferecido pela antecipação de tarefas, apenas

modificando o modelo de definição de processos. Este tipo de antecipação será chamado aqui de antecipação gerenciada *a priori,* pois ocorre em tempo de definição do processo de *workflow*, antes da execução.

A maneira de possibilitar a antecipação gerenciada *a priori* permitindo que haja troca de resultados intermediários é modelar as tarefas que podem ser antecipadas numa granularidade menor. Não se trata de uma nova abordagem, mas sim de uma nova prática de modelagem utilizando o modelo tradicional, sem a necessidade de novos elementos de modelagem ou novos conceitos.

Nesta nova prática de modelagem as tarefas antecipáveis e suas antecessoras devem ser "quebradas" em tarefas menores, ficando estas em parte seqüenciais e, em parte paralelas entre si. A Figura 4.2 ilustra como ficaria o modelo para o processo de concepção do artigo científico prevendo a antecipação de uma tarefa já durante a modelagem.

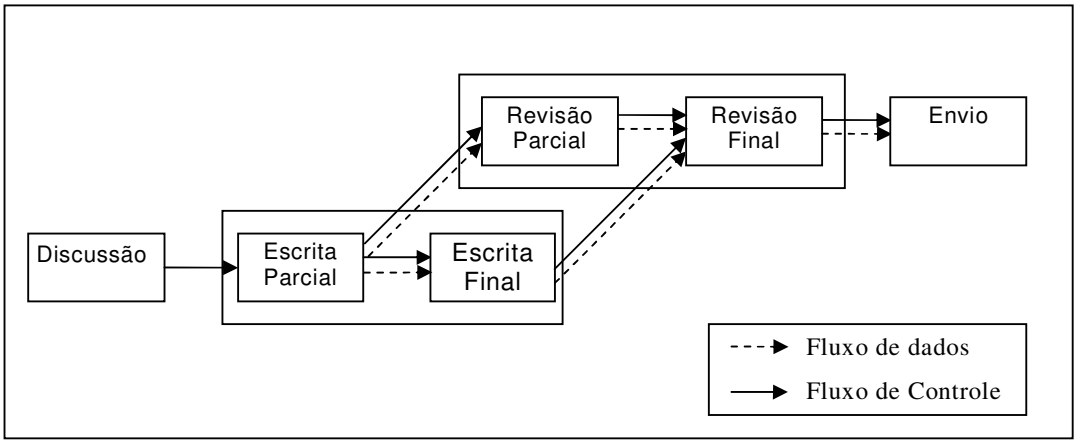

Figura 4.2: Antecipação utilizando modelo tradicional de definição de processos utilizando menor granularidade na modelagem das tarefas

Como pode ser visto, através da modelagem das tarefas em outro nível de granularidade é possível pré-determinar que tarefas sejam antecipadas. Neste caso o comportamento é o mesmo apresentado na Figura 4.1, deixando que as tarefas de escrita e revisão sejam executadas parcialmente em paralelo. Não é necessário utilizar nenhum tipo de elemento ou conceito adicional para este tipo de modelagem. Também não se faz necessário qualquer alteração na máquina de *workflow*. Qualquer ferramenta de definição de processos é capaz de modelar tal situação, e qualquer máquina de *workflow* pode interpretar, executar e gerenciar o referido processo.

O gerenciamento da antecipação *a priori* pode ser provido de duas maneiras distintas:

- **Manual**

  Os *workflow designers* são responsáveis por identificar e modelar as tarefas já na granularidade com que estas devem ser instanciadas e executadas. Uma tarefa que pode ser antecipada deve ter suas antecessoras divididas em "sub-tarefas", prevendo a maneira que deve se dar a antecipação e quantos

resultados intermediários (se houverem) serão enviados para as tarefas antecipáveis.

- **Automática**:

  Ao modelar um processo que possui tarefas que possam ser antecipadas, estas devem ser apenas identificadas pelo *workflow designer* responsável. Quando da instanciação deste processo, ele passará por um mecanismo de transformação que irá "quebrar" as tarefas identificadas, gerando-as numa granularidade maior. Tal granularidade poderá ser definida no momento da modelagem (através de um atributo extra na tarefa), ou no momento da transformação do modelo (através de um parâmetro passados ao mecanismo de transformação). No caso da definição do processo estar representada em XPDL, pode-se adotar um script XSLT precedendo a máquina de *workflow* que ficaria responsável por gerar o uma nova definição em XPDL. Esta nova definição conteria todas as tarefas antecipáveis já na nova granularidade especificada.

Quando tarefas antecipáveis são definidas manualmente não é necessário nenhum tipo de alteração no modelo tradicional de definição de processos. Apesar de possuir esta grande vantagem, é exigido muito esforço quando da modelagem dos processos.

Quando se utiliza a abordagem *a priori* automática é necessário que se modifique o modelo de definição de processos através da inserção de um identificador de tarefas antecipáveis. Uma vantagem de usar o mecanismo automático é a possibilidade de utilizar a definição de processos sem a mudança da granularidade. Porém, esta abordagem exige esforço de implementação de um pré-processador que faça a transformação do processo.

A grande vantagem de buscar a antecipação de tarefas através da alteração da granularidade é a não necessidade de modificações da máquina de *workflow*, nem da inserção de novos elementos de modelagem. Utilizando abordagem *a priori*, o fluxo transcorreria exatamente como descrito no modelo. O fluxo de controle funcionaria apenas com dependências do tipo *fim-início*, ficando a dependência com relação ao fluxo de dados condicionada ao final da tarefa. Em resumo, as dependências de dados e controle continuariam unificadas, sendo o início de uma tarefa condicionada ao final de suas antecessoras, estando a "produção" e o "consumo" dos resultados condicionados a esta transição.

No processo ilustrado na Figura 4.2, o dado de saída da tarefa "*Escrita Rascunho"* seria na verdade um resultado intermediário da tarefa "*Escrita"*. Quando tal resultado fosse enviado, o sistema daria a tarefa como concluída e dispararia suas sucessoras. Tal resultado serviria como entrada da tarefa "*Revisão"* (*Revisão Rascunho*) e da tarefa "*Escrita Final"*. Já a tarefa "*Revisão Final"* seria disparada quando da conclusão e envio de resultados finais da tarefa "*Escrita Final"* e da "conclusão" da tarefa "*Revisão Rascunho"*, que poderia ter como pós-condição a conclusão da tarefa "*Escrita Final"*.

Apesar de apresentar-se simplificada, a abordagem *a priori* continua produzindo processos rígidos. O grande problema apresentado pela abordagem é que, por tratar-se apenas de uma nova prática de modelagem, as tarefas continuam sendo caixas pretas para os agentes. A flexibilidade oferecida é pré-determinada e

está relacionada ao número de trocas estão previstas no modelo. No caso do exemplo (Figura 4.2), pode-se verificar que a tarefa "*Escrita*" está divida em duas outras, assim como a tarefa "*Revisão*". Fica então pré-definido que haverá uma e somente uma troca de resultados entre as tarefas, sendo flexível ao agente apenas o momento em que este irá fornecer tal resultado. Outros resultados poderiam ter sido enviados após este primeiro, porém, o modelo não permite tais interações.

Outro ponto que pode ser considerado negativo nesta abordagem é o excessivo gasto com modelagem dos processos. É muito complexo modelar este tipo de interação prevendo não só quais as tarefas que podem ser antecipadas, mas também, quantas vezes será feita tal troca de resultados. A modelagem de tais tarefas em um processo de médio ou grande porte exigiria muito esforço adicional. Tal esforço pode ainda ser incrementado com problemas quando da instanciação destes processos, pois, cada instância de um processo pode ter características totalmente diferentes, podendo ser uma tarefa antecipável em uma instância do processo e não antecipável em outra.

Resumindo, o gerenciamento *a priori* da antecipação tem a grande vantagem de poder ser utilizado em qualquer SGWf existente atualmente, sem a necessidade de esforço para implementação da máquina de *workflow*. Porém apresenta flexibilidade pré-definida na modelagem e tarefas executando atomicamente.

Na próxima seção será apresentada estratégia que visa reduzir o esforço de modelagem e aumentar a flexibilidade: a antecipação com gerenciamento *a posteriori*, na qual a antecipação está ligada basicamente ao tempo de execução de processos de *workflow*.

4.2 Antecipação Gerenciada *a posteriori*

Tendo em vista a dificuldade de prever com precisão a antecipação em tempo de definição de processos, são apresentadas alterações na máquina de *workflow* que a permitam gerenciar a antecipação em tempo de execução. Neste trabalho este tipo de abordagem é chamada *a posteriori*.

Esta abordagem foi construída com base na proposta por Grigori (2001), que prevê alteração nos estados das tarefas para permitir a antecipação. Porém, aqui são explicitados meios para identificar as tarefas antecipáveis e como isto é mapeado para a máquina de *workflow*. Outra contribuição ao modelo é a maneira com que são tratados as dependências de dados e controle e o mapeamento destas para o diagrama de estados das instâncias de tarefas.

No presente trabalho busca-se seguir os padrões e recomendações da WfMC, sendo toda a estratégia de antecipação baseada em tais padrões.

4.2.1 Identificação das Tarefas Durante a Definição de Processos

Uma grande facilidade seria prover antecipação de tarefas sem a necessidade de alteração alguma na definição do processo. Contudo, existe a necessidade de identificar, quando da definição do processo, aquelas tarefas que estão e aquelas que não estão autorizadas a ser antecipadas. Isto porque a antecipação pode não ser aplicável a todos os processos nem a todas as tarefas de um determinado processo. Existem processos e tarefas que devem ser executadas obrigatoriamente de maneira atômica e isolada.

Por exemplo, tarefas executadas por recursos não humanos (executadas automaticamente), possivelmente não saberiam como lidar com um resultado intermediário, não podendo ser disparadas. Existem ainda tarefas executadas por recursos humanos cuja execução deve, obrigatoriamente, ser iniciada quando da conclusão de suas antecessoras. Um exemplo deste tipo de tarefa é apresentado na Figura 4.2: a tarefa "Envio" não pode ser antecipada por ser obrigatoriamente atômica, devendo o artigo final ser submetido como um todo e não em partes. Então, tal tarefa seria definida durante a modelagem como não antecipável, ficando esta sujeita às condições de disparo tradicionais.

Então, é evidente a necessidade de adicionar uma informação às tarefas, indicando a possibilidade destas serem ou não antecipadas. Essa necessidade, porém, esbarra em alguns problemas apontados em outras abordagens (capítulo 3), que são o aumento da complexidade na definição dos processos e a alteração do modelo de definição de processos tradicional.

Para minimizar os problemas com a complexidade e não propor alterações no modelo tradicional, a alternativa encontrada foi a adição de um atributo às tarefas. Através deste atributo, o sistema poderia autorizar, ou não, o disparo antecipado das tarefas. O atributo proposto é de caráter não obrigatório, deixando a critério do gerente responsável pela criação do processo a decisão de utilizar a antecipação ou não.

Como estão sendo seguidos os padrões e recomendações estabelecidas pela WfMC, este atributo está previsto como um *ExtendedAttribute* presente no XPDL. Os detalhes de implementação serão apresentados no capítulo 6.

Com o uso de um novo atributo, qualquer ferramenta de modelagem que obedeça aos padrões propostos pela WfMC poderia facilmente definir processos com antecipação prevista. Além disso, este atributo não prende a definição de processo a uma única máquina de execução. Qualquer máquina de *workflow* que estiver seguindo os padrões WfMC entenderá a definição de processo, evidentemente sem considerar a antecipação de tarefas.

4.2.2 Flexibilizando o fluxo de controle

Tradicionalmente as tarefas relacionam-se através de uma dependência onde uma tarefa somente é iniciada quando da conclusão de suas antecessoras (dependência *fim-início*). No entanto, para possibilitar a antecipação foi necessário identificar e analisar outras possíveis alternativas de dependência de fluxo de controle. Além da tradicional *fim-início* (tarefas não antecipáveis) é apresentado um conjunto de tipos de dependências baseadas apenas no início e no fim da execução das tarefas.

Algumas convenções utilizadas nesta seção e no decorrer do trabalho são apresentadas a seguir. A tarefa em foco será tratada de *t*. A toda a tarefa *t* estão relacionadas uma ou mais antecessoras diretas (*t_ant*), contidas no conjunto *T_ANT(t)*. Da mesma maneira, *T_SUC(t)* representa o conjunto das tarefas que são sucessoras diretas (*t_suc*) de uma determinada tarefa *t*. Por fim, *inicio(t)* denota o instante de início da tarefa *t* e *fim(t)* representa a conclusão desta tarefa *t*.

As dependências identificadas e propostas para flexibilizar o fluxo de controle são apresentadas a seguir:

- *Dependência Forte*: este é o tipo de dependência padrão, isto é, é a dependência *fim-início*. Define a tarefa como *não antecipável,* fazendo-a obedecer ao fluxo de controle de maneira tradicional. Uma tarefa condicionada a este tipo de dependência deve aguardar que suas antecessoras sejam concluídas para iniciar sua execução no caso ser antecedida por um *AND-join*; ou uma de suas antecessoras seja concluída no caso de um *OR-join*. A regra que deve ser aplicada para que uma tarefa *t* obedeça este tipo de dependência é:

  $\forall t_ant \in T_ANT(t), (início(t) > fim(t_ant))$,

  no caso de *t* ser antecedida por um *AND-join* ou por uma única tarefa*;* e

  $\exists t_ant \in T_ANT(t), (início(t) > fim(t_ant))$,

  no caso de *t* ser antecedida por um *OR-join.*

- *Dependência Média:* esta é a dependência mais conservadora dentre as alternativas aqui propostas para definir uma tarefa como *antecipável*. Uma tarefa sujeita a este tipo de dependência somente poderá ser disparada após o início de todas as suas antecessoras, no caso de um *AND-join*. E, no caso de um *OR-join*, após o início de uma de suas antecessoras. Além disso, esta tarefa só poderá ser concluída ao final de suas antecessoras. Quando antecedida de um *AND-join* terá de aguardar a conclusão de todas as suas antecessoras. Caso seja um *OR-join*, é necessário aguardar que a tarefa que foi utilizada para satisfazer a condição de disparo seja concluída. A restrição que se aplica a este tipo de dependência é a seguinte:

  $\forall t_ant \in T_ANT(t), (início(t) > início(t_ant)) \wedge (fim(t) > fim(t_ant))$,

  no caso de *t* ser antecedida por um *AND-join* ou por uma única tarefa*;* e

  $\exists t_ant \in T_ANT(t), (início(t) > início(t_ant)) \wedge (fim(t) > fim(t_ant))$,

  no caso de *t* ser antecedida por um *OR-join.*

- *Dependência fraca*: São definidos dois tipos de dependência fraca distintas, de acordo com a restrição mantida:

  - *Início*: definindo-se uma tarefa com dependência de tipo *início-início* a torna antecipável. Isto flexibiliza o fluxo de controle de maneira a restringir o disparo desta tarefa ao início de suas antecessoras. Fica relaxada a dependência *fim-fim*, ficando a tarefa independente da conclusão de suas antecessoras. A regra que deve ser aplicada para que uma tarefa *t* obedeça este tipo de dependência é:

    $\forall t_ant \in T_ANT(t), (início(t) > início(t_ant))$,

    no caso de *t* ser antecedida por um *AND-join* ou por uma única tarefa*;* e

    $\exists t_ant \in T_ANT(t), (início(t) > início(t_ant))$,

    no caso de *t* ser antecedida por um *OR-join.*

- *Fim:* Este tipo de dependência não condiciona o disparo da tarefa (flexibiliza a dependência *início-início*). A flexibilidade oferecida por este tipo de dependência permite que uma tarefa seja iniciada a qualquer momento. A única restrição fica por conta da conclusão da tarefa. Para esta dependência fica valendo a seguinte restrição:

  $\forall t_ant \in T_ANT(t), (fim(t) > fim(t_ant)),$

  no caso de *t* ser antecedida por um *AND-join* ou por uma única tarefa; e

  $\exists t_ant \in T_ANT(t), (fim(t) > fim(t_ant)),$

  no caso de *t* ser antecedida por um *OR-join.*

- *Independente do Fluxo de Controle:* Nada restringe o disparo nem a conclusão das tarefas. As tarefas podem ser iniciadas a qualquer momento, estando todas disponíveis para execução. O disparo das tarefas estaria sujeito apenas aos dados de entrada da tarefa, se forem considerados fluxos de dados e de controle independentes.

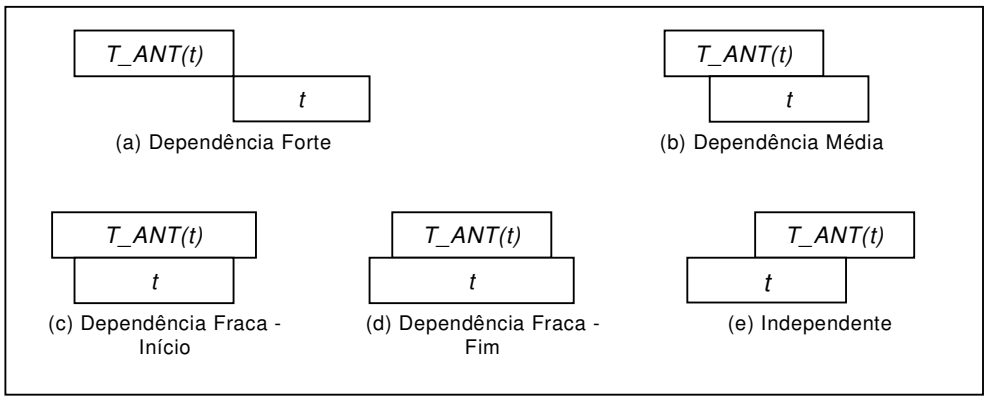

Figura 4.3: Tipos de dependência entre tarefas previstas

É válido lembrar que as tarefas estão ainda condicionadas ao fluxo de dados, estas dependências apresentadas são relativas apenas ao fluxo de controle. Assim, o disparo das tarefas está sujeito ainda à disponibilidade de seus dados de entrada.

Para o restante do trabalho será considerada que uma tarefa antecipável obedecerá à "*dependência média*", que oferece maior segurança e limita a antecipação. Isso diminui a chance de desperdício de trabalho e de re-trabalho. Apesar da possibilidade de escolher a dependência à qual uma tarefa resultar num aumento de flexibilidade durante a execução, isto aumentaria a complexidade da modelagem.

4.2.3 Fluxo de Dados

A fim de prover flexibilidade para o fluxo de dados, neste trabalho foram propostos novos estados para os elementos de dados: `inicial` e `nãoInicial`, sendo este último dividido em `intermediário` (`estável` ou `instável`) e

final. O diagrama de transição de estados proposto para os elementos de dados é apresentado na Figura 4.4, sendo detalhado em seguida.

Na abordagem tradicional, tentando traduzir em transição de estados o que ocorre com um elemento de dado, teria-se o seguinte resultado: ao ser solicitado por uma instância de tarefa, um elemento de dado entra no estado inicial, caso este elemento de dado torne-se uma saída da tarefa ele passa ao estado final, sendo enviado ao sistema somente quando da conclusão da tarefa. Os possíveis resultados intermediários produzidos seriam utilizados apenas dentro da própria instância da tarefa. Portanto, para que as dependências de dados de entrada de uma tarefa sejam satisfeitas, é necessário que seus dados de entrada sejam resultados finais de outras tarefas. Provendo meios para possibilitar o fornecimento de resultado intermediário pode-se alterar as pré-condições das tarefas a fim de permitir o início antecipado das mesmas.

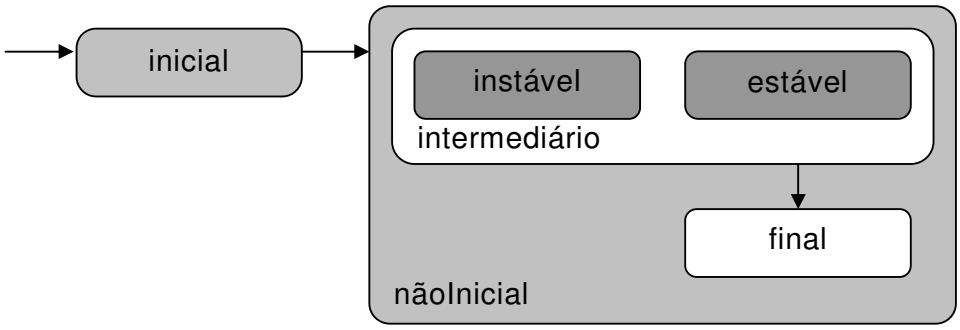

Figura 4.4: Diagrama de Transição de Estados proposto para os elementos de dados

Então, é proposto um conjunto de estados que permita um elemento de dado passar por um estado intermediário antes deste ser considerado final. É justamente este novo estado que permitirá o disparo de tarefas sem a necessidade dos dados de entrada das mesmas serem finais.

Todo o dado fornecido por uma instância de tarefa durante sua execução, que não for resultado final, será considerado intermediário. Um dado no estado intermediário pode ser classificado ainda como instável ou estável. Tal classificação pode parecer desnecessária uma vez que não se pode proibir que um agente altere um resultado anteriormente dito estável. Porém, esta informação é muito importante para possibilitar que os agentes fiquem cientes dos possíveis problemas que podem ocorrer quando do uso destes resultados intermediários.

Para exemplificar a diferença entre um resultado intermediário instável e um resultado intermediário estável será utilizado novamente o processo de produção de um artigo científico. Um agente responsável pela escrita de uma seção do artigo pode disponibilizar um primeiro rascunho de toda sua seção para revisão e correções. Desta maneira o documento pode ser editado tanto pelo editor quanto pelo revisor, passando a existir uma espécie de edição cooperativa (resultado intermediário instável).

Porém, o agente pode liberar uma subseção do artigo já numa versão "final" para revisão, garantindo que esta não mais será alterada, podendo o agente responsável pela revisão realizar sua tarefa sem preocupação com possíveis inconsistências (resultado `intermediário estável`).

No contexto deste trabalho um resultado `intermediário estável` é considerado pronto para ser utilizado como entrada para tarefas, isto é, pronto para ser modificado pelos outros agentes em outras tarefas, garantindo que este não mais será alterado na origem. Quando um agente A disponibiliza um de seus resultados e o classifica como `estável` este "perde" o direito de alterá-lo. Ao utilizar este resultado, um agente B tem plenos direitos sobre ele, podendo lê-lo e alterá-lo como desejar. Qualquer alteração realizada por A neste elemento de dado será desprezada quando outro resultado (`intermediário` ou `final`) for liberado, e houver a união das versões dos agentes A e B. No caso de conflitos entre as duas versões, a versão utilizada (ou alterada) por B será sempre considerada a versão válida.

Mecanismos de resolução de conflitos podem ser utilizados para lidar com os problemas inerentes à utilização dos resultados cooperativamente. Porém, estes mecanismos não serão aqui abordados. No presente trabalho um resultado `intermediário instável` continua sendo propriedade da tarefa de origem, podendo apenas ser consultado (*somente leitura*) pela(s) tarefa(s) de destino, sendo a versão válida aquela utilizada pela tarefa de origem. Ao contrário, um resultado dito `intermediário estável` torna-se propriedade da tarefa de destino assim que solicitado por esta. Apenas as alterações realizadas pela tarefa de destino serão consideradas quando da união das versões para a obtenção da versão final.

4.2.4 Modificando a Máquina de Execução

Para permitir que tarefas possam ser disparadas sem que suas condições sejam satisfeitas foi necessário alterar a máquina de *workflow*. É necessário que a máquina de execução possa identificar o novo atributo de identificação das tarefas e possa lidar com tipos diferentes de dependências de fluxo de controle e de fluxo de dados.

Toda a flexibilidade inerente a esta abordagem é inserida por alterações na máquina de *workflow*. Neste sentido, existem duas possibilidades. A primeira delas prevê a antecipação sem a necessidade de modificação na estrutura do processo, mas através da alteração no diagrama de estados das instâncias de tarefas. A outra diz respeito a alterações dinâmicas no modelo do processo, através da alteração da granularidade das tarefas em tempo de execução.

Neste trabalho o enfoque será dado às alterações no diagrama de estados das tarefas. A abordagem que utiliza a modificação dinâmica dos processos será apenas introduzida.

4.2.4.1 Antecipação através de alteração no diagrama de estados das tarefas

Esta abordagem é baseada na alteração da máquina de transição de estados do SGWf. A WfMC propõe um conjunto de estados padrão para os estados das instâncias das tarefas, permitindo detalhá-los de maneira a adequar-se a cada situação. O diagrama básico proposto pela WfMC é ilustrado na Figura 4.5. Os estados previstos na figura são descritos abaixo:

- `aberta`: a instância da tarefa está ativa;
  - `rodando`: a instância da tarefa está em execução;
  - `nãoRodando`: a instância está pronta, mas ainda não foi iniciada;
  - `suspensa`: a execução da instância da tarefa foi temporariamente suspensa, podendo ser retomada;
- `fechada`: execução da instância da tarefa foi finalizada;
  - `abortada`: execução da instância da tarefa foi abortada, provavelmente pelo aborto da instância do processo da qual esta fazia parte.
  - `terminada`: execução da instância da tarefa foi terminada, provavelmente pelo término da instância do processo da qual esta fazia parte. Uma instância de processo quando abortada, "mata" todas as instâncias de tarefas instantaneamente, enquanto quando terminada, as instâncias de tarefas correntes têm a possibilidade de serem concluídas.
  - `concluída`: a execução da instância da tarefa foi concluída normalmente, cumprindo todas suas pré-condições sem interferências externas.

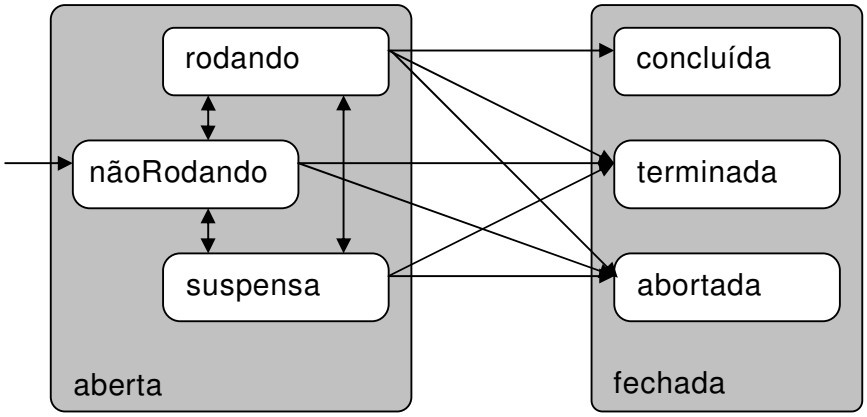

Figura 4.5: Diagrama de Estados proposto pela WfMC (WFMC, 1999)

Levando-se em conta que um SGWf tradicional utiliza este diagrama de transição de estados para as tarefas, obtém-se o comportamento descrito a seguir. Quando um processo é instanciado todas as tarefas entram em seu estado inicial, neste caso `nãoRodando`. A transição entre os estados `nãoRodando` e `suspensa` geralmente é disparada quando da suspensão da instância do respectivo processo. Para passar do estado `nãoRodando` para `rodando` é necessário que todos os pré-requisitos de execução da tarefa sejam satisfeitos e que a instância da tarefa seja alocada por um agente. Uma instância de tarefa que está `rodando`, pode ser `suspensa` através de solicitação do próprio agente, ou pela

suspensão da respectiva instância de processo. Quando a execução de uma instância de tarefa é finalizada, seu estado passará a ser um dos sub-estados de `fechada` (`concluída`, `terminada` ou `abortada`). A definição depende da maneira com que esta instância foi finalizada. O estado `concluída` só pode ser alcançado através do estado `rodando`, uma vez que ele representa a finalização normal de uma tarefa.

A fim de prover a flexibilidade quando da execução do processo, foi necessário, então, alterar o diagrama de estados das tarefas da máquina de execução do *SGWf*. Tal alteração é proposta a fim de tornar a antecipação clara e simples para o sistema. Para a concepção de tal diagrama, foram utilizados como base o diagrama de estados proposto pela WfMC (Figura 4.5) e o diagrama de estados previsto no *Coo-Flow* (GRIGORI; CHAROY; GODART, 2004), que leva em conta a antecipação de tarefas. O diagrama de estados resultante e seus respectivos estados são apresentados na Figura 4.6.

Neste novo diagrama são apresentados os estados `pronta` e `nãoPronta` (sub-estados de `nãoRodando`) e os estados `antecipando` e `executando` (sub-estados de `rodando`). O estado `suspensa` foi re-alocado como um sub-estado de `nãoRodando`, uma vez que, quando uma instância de tarefa está `suspensa`, obviamente, não está rodando. Em níveis de granularidade mais altos, foram definidos alguns outros estados. O estado `pronta` é dividido ainda em dois sub-estados: o estado `paraAntecipar` e `paraExecutar`. O estado `nãoPronta` faz distinção entre os sub-estados `antecipável` e `nãoAntecipável`. Alguns destes estados possuem mais um nível de aninhamento que serão detalhados durante a presente seção.

A seguir são apresentados os estados propostos e a descrição de cada um:

- `suspensa`: continua com o mesmo significado, indicando que uma instância de tarefa foi parada, porém pode ser retomada a qualquer instante;
- `nãoPronta`: é o estado inicial das instâncias. Indica que a tarefa foi instanciada, porém ainda não tem suas condições satisfeitas para ser disponibilizada para execução;
    - `antecipável`: indica que a instância da tarefa poderá ser antecipada se tiver suas condições de antecipação satisfeitas;
    - `nãoAntecipável`: indica que a instância da tarefa não poderá ser executada antecipadamente;
- `pronta`: a instância teve suas pré-condições (de execução ou antecipação) satisfeitas e está pronta para ser iniciada;
    - `paraAntecipar`: indica que a instância da tarefa teve as pré-condições de antecipação satisfeitas e aguarda ser iniciada;
    - `paraExecutar`: a instância pode ter sua execução iniciada, pois suas pré-condições de execução foram satisfeitas por completo;
- `rodando`: a instância foi solicitada por um agente e foi iniciada;

- o `antecipando`: tarefa está sendo rodada de maneira antecipada;
- o `executando`: tarefa está executando e pode ser concluída normalmente.

Quando se instancia um processo, todas suas tarefas são instanciadas e são levadas para seu estado inicial. Neste caso o estado inicial das instâncias de tarefa é `nãoPronta`. A instância é então mapeada para um dos dois sub-estados: `antecipável` ou `nãoAntecipável`. Tal classificação é realizada durante a fase de modelagem do processo, como descrito na seção 4.2.1.

A transição bidirecional existente entre estes dois estados deve-se à possibilidade de alteração desta classificação mesmo durante a execução do processo. Enquanto a instância da tarefa não passou ao estado `pronta` é possível modificar sua condição passando-a de `antecipável` a `nãoAntecipável` e vice-versa.

Possibilitando tal mudança de estado durante a execução aumenta-se a flexibilidade do sistema, viabilizando que agentes solicitem ao gerente a antecipação de tarefas modeladas como não antecipáveis. Pode ser permitido, inclusive, que os agentes verifiquem a possibilidade de utilizar ou fornecer resultados intermediários e executem a operação de mudança de estado. Além disso, é possível adequar cada instância de um processo para suas necessidades e características próprias apenas alterando o estado de uma tarefa, sem que seja preciso modificar o modelo do processo, ou evoluir o esquema do *workflow*.

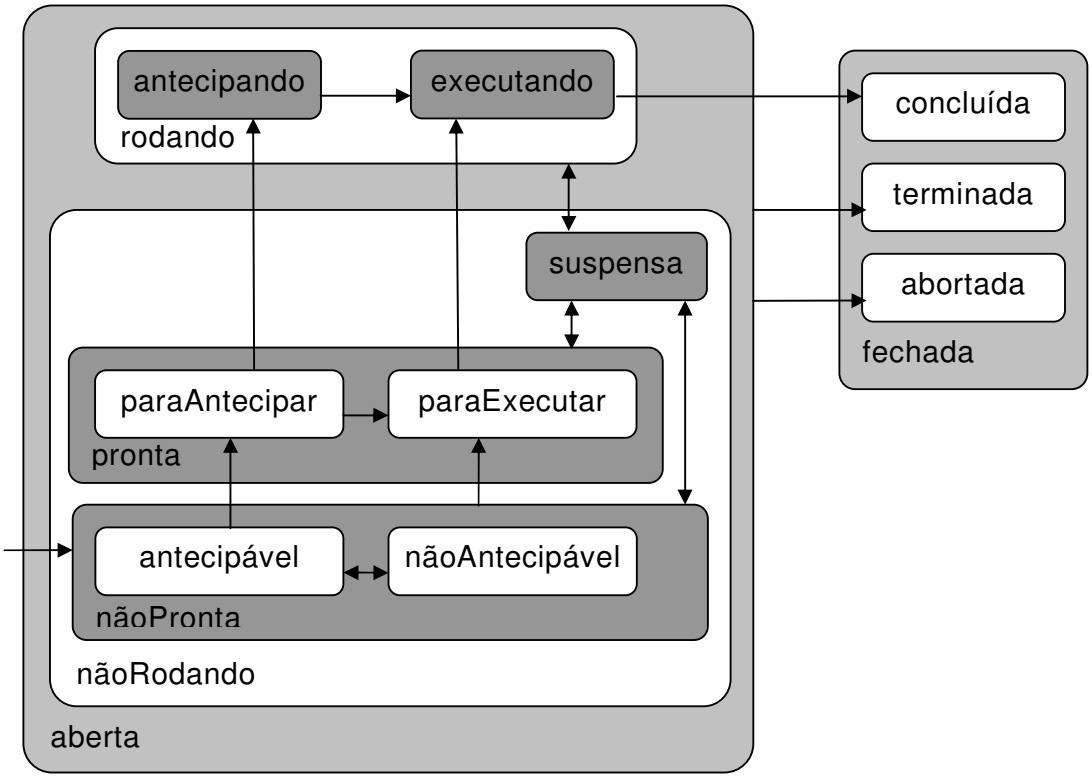

Figura 4.6: Diagrama de Estados para as Instâncias de Tarefas prevendo a antecipação

Para que uma instância de tarefa torne-se `pronta`, é necessário que se cumpram suas pré-condições com relação ao fluxo de controle e de dados. Então, os estados `antecipável` e `nãoAntecipável` foram divididos em sub-estados que representem estas condições. Os estados gerados a partir delas são apresentados na Figura 4.7 e referem-se a:

- *antecipação*: uma instância de tarefa é antecipável ou não antecipável (`antecipável`/`nãoAntecipável`);
- *fluxo de dados*: os dados de entrada obrigatórios daquela instância de tarefa estão disponíveis como resultados finais (`dadosOK`), disponíveis como resultados intermediários (`dadosInterm`) ou não estão disponíveis (`dadosNaoOK`); e
- *fluxo de controle*: a dependência de disparo daquela instância foi satisfeita por completo (`controleOK`), foi satisfeita de acordo com a *dependência média* (`controleInício`) ou não foi satisfeita ainda (`controleNãoOK`).

Como pode ser observado acima, com relação à antecipação existem dois possíveis valores, que deram origem aos sub-estados de `nãoPronta`. Quanto ao fluxo de controle são três valores, e com relação ao fluxo de dados mais três. Combinando-se todas as possibilidades de valores chega-se a um total de 18 estados. Estas combinações são apresentadas como estados no diagrama apresentado na Figura 4.7.

As transições entre estes novos estados é intuitiva. Entre os estados relativos ao fluxo de controle tem-se que uma instância de tarefa é iniciada no estado `controleNãoOk`. Quando todas suas antecessoras tiverem iniciado sua execução (estado `rodando`), esta instância passa ao estado `controleInício`. Já a transição entre `controleInício` e `controleOK` ocorre quando as instâncias antecessoras são concluídas.

Com relação ao fluxo de dados, as transições ocorrem de acordo com o dado fornecido. O estado inicial com relação ao fluxo de dados é `dadosNãoOK`. Se todos os dados de entrada de uma instância de tarefa estão disponíveis e são resultados finais, esta tarefa passa ao estado `dadosOK`. Se algum ou todos esses dados forem resultados intermediários, esta instância passa ao estado `dadosInterm`.

Na Figura 4.7 são apresentadas 13 das 18 combinações de condições previstas, porque existem 5 possibilidades em que ocorre a transição para um dos sub-estados do estado `pronta`. O estado `pronta.paraAntecipar` representa o preenchimento da condição `antecipável` combinada com: "`dadosInterm` e `controleInício`" ou "`dadosInterm` e `controleOK`" ou ainda "`dadosOK`, `controleInício`". Já o estado `pronta.paraExecutar` representa a satisfação das outras duas combinações de condições restantes: "`antecipável`, `dadosOK` e `controleOK`" ou "`nãoAntecipável`, `dadosOK` e `controleOK`".

O preenchimento das condições citadas no parágrafo anterior explica duas transições existentes no diagrama da Figura 4.6. A primeira delas é entre o estado

`nãoPronta.antecipável` e o estado `pronta.paraAntecipar`. Esta transição depende do cumprimento das condições de disparo antecipado (dependência média do fluxo de controle e presença de resultados intermediários).

A outra é a transição existente entre o estado `nãoPronta` e o estado `pronta.paraExecutar`. Esta tem como origem o estado `nãoPronta`, pois as únicas informações que interessam para ativar tal transição dizem respeito ao fluxo de dados e fluxo de controle. Se estas duas condições forem satisfeitas, a instância da tarefa passa a estar pronta para executar, não havendo restrições relativas à possibilidade de antecipação.

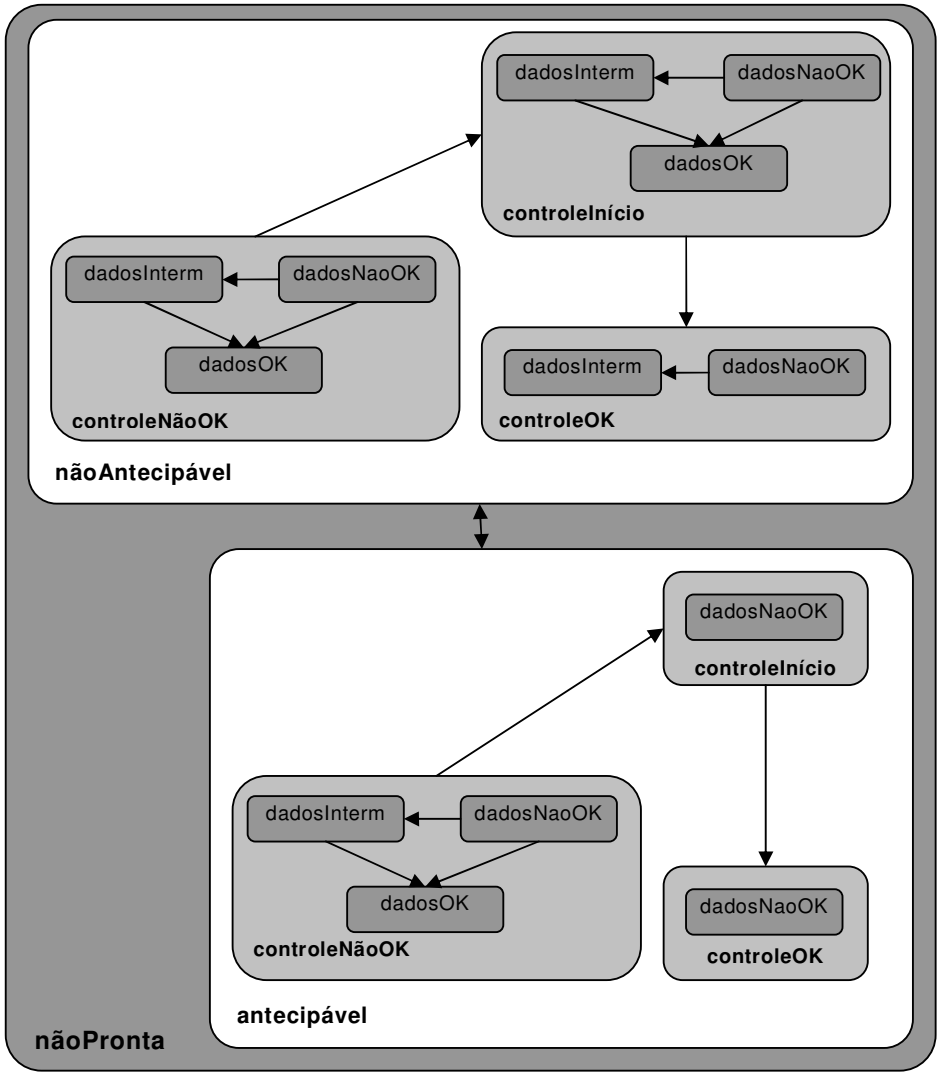

Figura 4.7: Especialização do Estado `nãoPronta` prevendo as condições de disparo

Do estado `pronta` uma instância de tarefa pode passar ainda para os estados `suspensa` e `rodando`. A transição para o estado `suspensa` é bidirecional e indica que uma tarefa pode ser pausada e voltar ao seu estado de origem mais tarde. Já as transições existentes para o estado `rodando` são disparadas quando

um dos agentes escolhe iniciar a execução de uma das instâncias de tarefa oferecidas em sua lista de tarefas. O destino da transição depende do seu estado de origem. Sendo assim, caso a tarefa estiver no estado `pronta.paraAntecipar` passará ao estado `antecipando`, caso esteja `pronta.paraExecutar` passará ao estado `executando`.

Quando estiver `rodando` de maneira antecipada (`antecipando`), a instância da tarefa necessita ter suas pré-condições de disparo satisfeitas a fim de entrar no estado `executando`. Portanto, uma instância de tarefa permanece no estado `antecipando` até que todos seus dados de entrada sejam resultados finais e que suas antecessoras estejam `concluídas`. Quando `executando`, uma tarefa pode ser, então `concluída`, isto é, pode ser finalizada de maneira normal.

Tabela 4.1: Combinação das condições de disparo e o estado que as representa

Fluxo de Controle	***Fluxo de Dados***	***Antecipação***	**Estado**
controleNãoOK	*dadosNaoOK*	*antecipável*	Sub-estado de Antecipável
controleNãoOK	*dadosNaoOK*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleNãoOK	*dadosInterm*	*antecipável*	Sub-estado de Antecipável
controleNãoOK	*dadosInterm*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleNãoOK	*dadosOK*	*antecipável*	Sub-estado de Antecipável
controleNãoOK	*dadosOK*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleInício	*dadosNaoOK*	*antecipável*	Sub-estado de Antecipável
controleInício	*dadosNaoOK*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleInício	***dadosInterm***	***antecipável***	***pronta.paraAntecipar***
controleInício	*dadosInterm*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleInício	***dadosOK***	***antecipável***	***pronta.paraAntecipar***
controleInício	*dadosOK*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleOK	*dadosNaoOK*	*antecipável*	Sub-estado de Antecipável
controleOK	*dadosNaoOK*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleOK	***dadosInterm***	***antecipável***	***pronta.paraAntecipar***
controleOK	*dadosInterm*	*nãoAntecipável*	Sub-estado de nãoAntecipável
controleOK	***dadosOK***	***antecipável***	***pronta.paraExecutar***
controleOK	***dadosOK***	***nãoAntecipável***	***pronta.paraExecutar***

4.2.4.2 Antecipação através da alteração dinâmica dos processos

Uma alternativa à alteração no diagrama de estados tem como base a alteração dinâmica de processos (REICHERT; RINDERLE; DADAM, 2003). Para o caso específico da antecipação de tarefas a operação que interessa é a de *inserção de tarefas* em tempo de execução. Esta inserção, neste caso, deve ser realizada de maneira automática.

A máquina de *workflow* fica responsável por identificar as tarefas antecipáveis e permitir o início antecipado quando suas pré-condições de antecipação forem cumpridas. Quando do cumprimento das condições o sistema "quebra" a tarefa de origem e suas sucessoras, gerando um fluxo paralelo e permitindo o início antecipado destas últimas. Um sistema que funcione com esta abordagem pode gerenciar múltiplos envios de resultados por uma mesma tarefa, aumentando cada vez mais a granularidade.

Nesta abordagem seriam utilizadas definições de processos idênticas às utilizadas na abordagem anterior (alterando o diagrama de estados). Porém, ao

final de cada instância pode-se obter um processo totalmente diferente, de acordo com o número de troca de resultados e de quais tarefas foram instanciadas.

A flexibilidade com relação ao fluxo de controle e de dados apresentadas anteriormente são igualmente aplicáveis a esta abordagem. Desta maneira a utilização da operação de *inserção de tarefas* ficaria restrita àquelas modeladas como antecipáveis.

Os benefícios conseguidos através desta alternativa devem ser similares aos obtidos com as alterações no diagrama de estados das tarefas. Porém, a complexidade das alterações da máquina de *workflow* são muito maiores. Isto devido à necessidade de um conjunto de regras que garantam a consistência do modelo após a modificação.

4.3 Percepção como apoio à Antecipação

O aumento na cooperação inerente à antecipação de tarefas pode influenciar negativamente o desempenho do processo, causando aumento também no esforço de comunicação exigido. Isto pode tornar o uso a antecipação uma estratégia não muito útil às organizações. Para tentar minimizar este tipo de problema podem ser utilizados elementos de percepção que suportem tal antecipação, de maneira a controlar o esforço inserido pelo aumento da cooperação, sem comprometimento do desempenho.

Nesta seção é apresentada uma breve discussão sobre elementos de percepção e sua possível utilidade no suporte à antecipação de tarefas. O objetivo destes elementos de percepção seria a redução do esforço de comunicação inserido pela possibilidade de antecipar tarefas. Se a antecipação fosse permitida sem que o sistema tomasse parte, muito tempo seria utilizado durante a comunicação entre os agentes envolvidos em tal antecipação.

Basicamente, o mecanismo de comunicação necessário para permitir o disparo antecipado de uma tarefa teria quatro passos:

1. Formulação do pedido de resultado intermediário para permitir o disparo de sua tarefa;
2. Envio do pedido aos agentes responsáveis pelas tarefas antecessoras;
3. Recebimento, entendimento e verificação da possibilidade de atender à solicitação do pedido;
4. Formulação e envio da resposta ao agente que solicitou o resultado para dar início à antecipação.

Então, a cada vez que for desejado o início antecipado de uma tarefa, é necessário o cumprimento destes quatro passos, no mínimo. Isso tomaria tempo tanto de quem fornece o resultado como de quem solicita.

Outro momento crítico é o momento de envio de uma outra versão de um dado que já está sendo utilizado por outras tarefas. Nesse caso, seria necessário que fossem enviadas mensagens avisando a todos os agentes que estão utilizando este dado que um novo resultado foi carregado para o repositório central. Seria inviável e muito custoso que o agente enviasse mensagem aos interessados a cada vez que

fizesse uma atualização. Existe ainda a possibilidade de existirem conflitos de versões durante estes envios, necessitando informar a todos os agentes envolvidos para possibilitar a resolução. Isso tudo não só pode trazer grandes perdas para o processo, como pode sobrecarregar certos usuários com troca de mensagens.

Fica claro que o esforço adicional de comunicação necessário à antecipação de tarefas pode inviabilizar seu uso. Com um mecanismo capaz de conscientizar os agentes da execução do processo, do estado dos dados e das possibilidades inerentes às suas tarefas, tal aumento no esforço pode ser minimizado.

Como base para todo o mecanismo devem ser utilizadas listas de trabalho (*worklists*), que é o elemento de percepção mais comum em SGWfs. A partir destas listas, os agentes têm acesso a todas as outras informações relativas às tarefas, como datas relevantes, descrição, nome, estado da tarefa, entre outros. Este tipo de percepção é chamado *process awareness* (GODART *et al.*, 2004) ou Percepção de Tarefas (ARAÚJO, 2000).

Com relação ao andamento do processo é necessário prever os novos estados inerentes à antecipação, deixando os agentes cientes do modo como estão sendo executadas as tarefas dentro dos processos. É importante ao agente saber, por exemplo, se existem tarefas prontas para antecipar, ou se os dados que estão disponíveis para a antecipação de uma tarefa vêm de uma tarefa que está executando ou antecipando. Conhecer as dependências das tarefas é de grande importância para um bom andamento de um trabalho cooperativo (ARAÚJO, 2000).

O mecanismo de percepção deve fornecer elementos que motivem os agentes a antecipar, e também que motivem o fornecimento de resultados intermediários. Neste sentido os agentes devem ser informados sobre a necessidade e a possibilidade de seus sucessores serem antecipados, observando-se a disponibilidade dos agentes responsáveis pelas tarefas sucessoras e a possibilidade destas serem antecipadas (antecipáveis/não antecipáveis). Com estas informações, são identificadas as tarefas que são "potenciais fornecedoras de resultados intermediários". Estas podem ser alertadas e motivadas a disponibilizar seus resultados parciais.

É necessário também manter os agentes cientes das condições dos dados através de elementos de percepção de estado (*state awareness*) (GODART *et al.*, 2004). Estes elementos têm por objetivo mostrar toda e qualquer alteração que ocorrer com os dados relacionados às tarefas que estão sendo executadas pelo agente. Desta forma os agentes podem verificar se existe uma nova versão de um dado ou se existe algum conflito relacionado a seus elementos de dados, sem precisar entrar em contato com os outros agentes.

Seria interessante se tais informações pudessem ser acessadas de maneira progressiva, de forma a não sobrecarregar a interface dos agentes. Em uma primeira instância podem ser apresentados apenas ícones/cores para apresentar o estado, condições e possibilidades relacionadas às tarefas. Se desejar maiores explicações sobre aquelas informações, os agentes podem seguir acessando as informações, navegando através dos elementos de percepção.

Ainda com relação à sobrecarga, deve ser permitido ao agente escolher as informações que deseja receber em sua lista de trabalho, e ir ajustando de acordo

com a utilização de cada um dos elementos. Isto traz flexibilidade em outro ponto da execução de um processo de *workflow,* a interface com o usuário.

Alguns elementos de percepção social (ARAÚJO, 2000) também podem ser inseridos possibilitando aos agentes perceberem quais outros usuários estão cooperando com suas tarefas. Este tipo de percepção deve oferecer elementos que mostrem a disponibilidade e também meios para que seja possível a comunicação entre os membros de maneira simples.

Todos os elementos aqui analisados e apontados são apresentados na Tabela 4.2. Nesta tabela são apresentadas as informações que devem ser apresentadas e uma explicação detalhada dos elementos que podem ser apresentados. Vale lembrar que na tabela estão presentes aqueles elementos considerados úteis à antecipação. Isto não quer dizer que os elementos não citados não são úteis, eles apenas não necessitam alteração ou não contribuem para a antecipação de tarefas.

Tabela 4.2: Tipos de percepção e sua importância para o suporte à antecipação de tarefas

Tipo de percepção	**Informação apresentada**	**Detalhes**
Percepção de Estado	Estado dos elementos de dados relacionados às tarefas do agente	Mostrar se um elemento é atual, se há nova versão ou se existem possíveis conflitos de versão a serem resolvidos.
		Trazer facilidades para resolução de conflitos.
Percepção de Processo	Status e Andamento das Tarefas	Apresentar o estado das tarefas disponíveis para o agente (executável, antecipável, não pronta, ...)
		Indicar atrasos e prazos.
		Apresentar meios simples de solicitar disparo e conclusão das tarefas.
		Apresentar alertas que informem e motivem a antecipação.
	Estrutura do processo	Apresentar as dependências e os relacionamentos entre as tarefas.
		Se rodando, mostrar as tarefas que estão aguardando sua conclusão ou resultados intermediários.
		Se não rodando, mostrar as tarefas das quais esta está dependendo ou aguardando resultados.
		Alertar os "potenciais fornecedores de resultados intermediários" da necessidade de seus resultados para o disparo outras tarefas.
Percepção Social	Proximidade/Cooperação	Apresentar os agentes que estão cooperando ou que possuem ligação com as tarefas que estão sendo executadas.
		Fornecer meios que permitam a comunicação entre os membros da equipe.
	Disponibilidade	Mostrar se os membros da equipe estão disponíveis/indisponíveis/ocupados.

Através da implementação correta do mecanismo de percepção com o conjunto de elementos aqui propostos é possível reduzir o esforço de comunicação introduzido pela antecipação de tarefas. Se funcionais, estes elementos podem maximizar os ganhos com relação ao tempo de execução dos processos quando da utilização da antecipação.

É de grande importância a realização de uma análise da utilização destes elementos em um ambiente real. Porém, a implementação e análise dos resultados da utilização dos elementos de percepção não são abordados na presente dissertação. Estes estudos estão sendo executados paralelamente a este trabalho e serão implementados no protótipo de sistema gerenciador de *workflow* alterado para suportar antecipação.

4.4 Considerações Finais

Este capítulo especificou a antecipação de tarefas que foi tratada na presente dissertação, suas vantagens e como ela pode ser gerenciada através de sistemas de *workflow*.

Uma das maneiras de antecipação apresentada é aquela cujo gerenciamento ocorre em tempo de definição, chamada aqui *a priori*. Neste caso a antecipação é totalmente definida antes da instanciação dos processos, tratando-se de uma prática de modelagem diferenciada, onde as tarefas são definidas em uma granularidade menor.

A grande vantagem deste tipo de antecipação é a possibilidade de utilizá-la em qualquer SGWf existente. Isto significa que se evita ter que alterar a máquina de *workflow*, que é um esforço considerável, visto a complexidade da referida máquina. Porém, a flexibilidade oferecida é pré-determinada, uma vez que as tarefas continuam sendo atômicas.

A outra abordagem especificada aumenta a flexibilidade oferecida, fazendo o gerenciamento *a posteriori* da antecipação, isto é, em tempo de execução dos processos. A abordagem deste tipo apresentada neste capítulo tem base no diagrama de estados proposto por Grigori (2001) e define meios de antecipar tarefas através da flexibilização dos fluxos de controle e dados.

A fim de identificar as tarefas antecipáveis e coibir a antecipação de outras tarefas, aqui se propõe a identificação em tempo de definição dos processos das tarefas que podem ser iniciadas de maneira antecipada. Isto é feito através de um atributo extra colocado nas tarefas (no caso, um *ExtendedAttribute* previsto no XPDL). Caso não sejam definidas estas tarefas, existe flexibilidade suficiente para que, em tempo de execução, o gerente faça tal definição, ou permita que os agentes a faça.

O fluxo de controle é flexibilizado através de novas dependências intertarefas. Alguns possíveis relacionamentos que possibilitam a antecipação foram especificados. A fim de reduzir o re-trabalho e a perda de trabalho, optou-se por utilizar-se a *dependência média* para representar a dependência relativa à antecipação de tarefas. Quanto ao fluxo de dados fica permitido o uso "compartilhado" de resultados intermediários entre tarefas seqüenciais, permitindo que duas ou mais tarefas cooperem utilizando um mesmo elemento de dado.

Estas novas dependências foram mapeadas para o novo diagrama de estados das instâncias de tarefas. Cada uma das dependências (de dados e de controle) representa um conjunto de estados capazes de prever o disparo antecipado das tarefas.

A flexibilidade oferecida por este tipo de antecipação é muito maior se comparada à abordagem *a priori*, não obrigando a antecipação e não definindo o número de interações que deve haver durante a execução do processo. Além disso, nesta abordagem o esforço necessário durante a modelagem é menor que na abordagem *a priori*.

A desvantagem da utilização da antecipação gerenciada *a posteriori* fica por conta do esforço necessário para alterar da máquina de *workflow*.

A abordagem *a posteriori* foi adotada na implementação do protótipo de simulador aqui proposto, da ferramenta de definição de processos Amaya *Workflow* (TELECKEN, 2004) e da máquina de *workflow* de França (2004), que serão apresentadas no capítulo 6.

Além das abordagens apresentadas, ainda são analisados possíveis elementos de percepção que dêem suporte à antecipação, visando reduzir o esforço de comunicação inerente à cooperação. Este ponto encontra-se em aberto, sendo aqui apresentada apenas uma discussão inicial e poucas definições sobre o assunto.

No próximo capítulo são apresentadas as simulações realizadas que mostram o desempenho do uso da antecipação de tarefas levando em conta o tempo de execução dos processos.

5 SIMULAÇÕES

A simulação por computador é um eficiente meio de estudar processos e prever seu desempenho (LAGUNA; MARKLUND, 2004). Com ela, pode-se analisar vários aspectos de um processo de maneira quantitativa, fornecendo um panorama dinâmico de seu funcionamento (TRAMONTINA, 2004). Segundo Aiello (2004) através de simulações os desenvolvedores de processos podem executar um processo em diferentes cenários enquanto variam os valores e distribuições de entrada. Ao final da simulação, as ferramentas podem prover informações valiosas sobre o processo. Torna-se então uma valiosa ferramenta para o estudo de processos de *workflow* quando se necessita obter resultados numéricos.

Com a finalidade de verificar o desempenho da utilização da antecipação de tarefas foram realizadas algumas simulações. O resultado de tais simulações traz, em termos numéricos, um comparativo entre a utilização e a não utilização da antecipação de tarefas. A métrica utilizada tem como base o tempo de execução dos processos.

Para possibilitar tais simulações foi implementado um protótipo de simulador específico, capaz de simular a característica de antecipação em processos de *workflow*. Este protótipo é apresentado no capítulo 6. Neste capítulo serão apresentados as simulações realizadas e os resultados obtidos.

As simulações aqui apresentados representam a contribuição mais significativa desta dissertação. Elas utilizam a abordagem *a posteriori,* tendo como base os diagramas de transição de estados apresentados no capítulo 4.

5.1 Cenário Utilizado

Para exemplificar a simulação que será realizada com o protótipo implementado, foi utilizada uma definição de processo básica de escrita de artigo científico. Este processo é apresentado na Figura 5.1 e é o mesmo que foi utilizado nas simulações. O processo em questão contém as características que se deseja tratar neste trabalho: é um processo predominantemente intelectual, com tarefas que podem ser antecipadas, troca de resultados entre tarefas e fluxos seqüenciais e paralelos.

Na definição de processo apresentada existem algumas informações adicionais. As tarefas sombreadas são aquelas tidas como antecipáveis. Os valores mostrados sobre as tarefas correspondem às informações de duração mínima e duração máxima de cada uma delas. Estas informações são apresentadas, junto do tempo médio de execução de cada uma das tarefas, na Tabela 5.1.

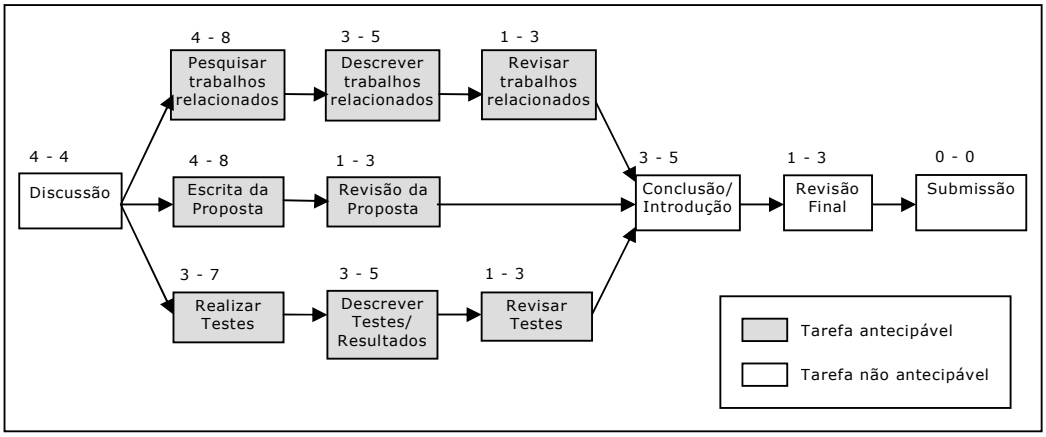

Figura 5.1: Processo criado para simulação

Tabela 5.1: Duração Mínima, Máxima e Média das tarefas

Tarefa	*Duração Mínima*	*Duração Máxima*	**Duração Média**
Discussão	4	4	4
Pesquisar trabalhos relacionados	4	8	6
Escrita da proposta	4	8	6
Realizar testes	3	7	5
Descrever trabalhos relacionados	3	5	4
Revisão da proposta	1	3	2
Descrever Testes/Resultados	3	5	4
Revisar trabalhos relacionados	1	3	2
Revisar testes	1	3	2
Conclusão/Introdução	3	5	4
Revisão Final	1	3	2
Submissão	0	0	0

A seguir o processo ilustrado será utilizado para exemplificar a simulação. Será realizada então uma demonstração da execução do processo utilizando a abordagem tradicional (sem antecipação) e utilizando a antecipação.

5.1.1 Execução Utilizando a Abordagem Tradicional

A partir deste momento, será apresentada uma ocorrência específica do processo modelado na Figura 5.1 se fosse utilizada a abordagem tradicional de execução de um processo de *workflow*. Para tal serão feitas algumas considerações:

- O tempo de execução de cada tarefa é o tempo médio previsto para cada uma delas. Estes valores são apresentados na Tabela 5.1.
- O tempo de execução é dado em dias.
- Uma tarefa inicia sua execução imediatamente após o término de sua(s) antecessora(s) direta(s). Isto é, assim que uma tarefa tem suas pré-condições satisfeitas ela passa ao estado *executando*.
- No caso do processo apresentado, existe um *AND-split* e um *AND-join*. O término da tarefa anterior ao *AND-split* dispara todas as

tarefas que o sucedem. Já a tarefa que sucede o *AND-join* somente é disparada quando todas as suas antecessoras forem concluídas.

Seguindo o fluxo apresentado, a *dependência forte* entre as tarefas e os tempos de execução propostos, o processo terá seu objetivo alcançado no instante "22", com a submissão do artigo final. As informações referentes à execução do processo podem ser vistas na Tabela 5.2 e, graficamente, na Figura 5.2.

Tabela 5.2: Comportamento das tarefas durante a execução do processo de maneira tradicional

Tarefa	*Início*	*Conclusão*	*Duração*
Discussão	0	4	4
Pesquisar trabalhos relacionados	4	10	6
Escrita da proposta	4	10	6
Realizar testes	4	9	5
Descrever trabalhos relacionados	10	14	4
Revisão da proposta	10	12	2
Descrever Testes/Resultados	9	13	4
Revisar trabalhos relacionados	14	16	2
Revisar testes	13	15	2
Conclusão/Introdução	16	20	4
Revisão Final	20	22	2
Submissão	22	22	0

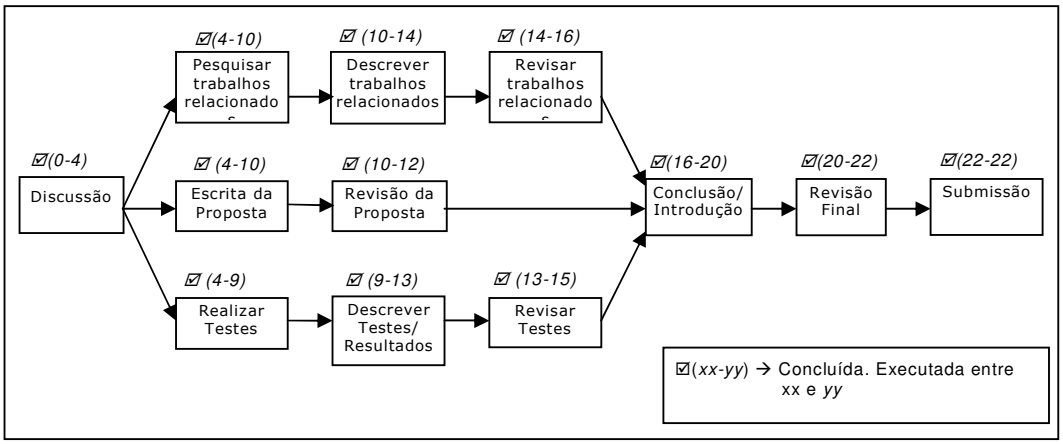

Figura 5.2: Processo executado utilizando a abordagem tradicional

5.1.2 Execução Utilizando Antecipação de Tarefas

O mesmo processo será utilizado para exemplificar a ocorrência utilizando a antecipação de tarefas. Para este exemplo também serão feitas algumas considerações:

- O tempo de execução de cada tarefa é o tempo médio de execução previsto para cada uma delas.
- O tempo de execução é dado em dias.

- Todas as tarefas antecipáveis obedecem à dependência de fluxo de controle do tipo *média* (seção 4.2.2). Devido à dependência com relação à conclusão das tarefas, *fim(t)* somente é possível um dia após *fim(t_ant)* se concretizar.
- Uma tarefa libera um resultado intermediário sempre que atingir 50% do seu tempo de execução. Isto é, quando todas as antecessoras de uma tarefa antecipável atingirem 50% de sua execução esta poderá iniciar a antecipação.
- Uma tarefa inicia sua execução imediatamente após o término de sua(s) antecessora(s) direta(s). Isto é, assim que uma tarefa tem suas pré-condições satisfeitas ela passa ao estado *executando*.
- O mesmo acontece com tarefas antecipáveis, que iniciam sua antecipação assim que suas pré-condições forem satisfeitas (no caso, 50% de suas antecessoras estarem concluídas). Isto é, assim que uma tarefa tem suas pré-condições de antecipação satisfeitas ela passa ao estado *antecipando*.

Ao instanciar-se o processo proposto, todas as tarefas passam ao estado *nãoRodando*, porém, sendo direcionadas para seu respectivo sub-estado de acordo com a definição do processo (*antecipável* ou *não antecipável*).

A Tabela 5.3 apresenta o andamento da execução do processo utilizando antecipação. A Figura 5.3 ilustra o processo com seus respectivos valores de tempo de disparo e de conclusão. Com a antecipação o processo é concluído em apenas 16 dias.

Com relação à execução, alguns esclarecimentos são necessários. As tarefas "Revisão da Proposta", "Revisar trabalhos relacionados" e "Revisar testes" apresentam tempo de execução maior que o previsto. Isto se deve à restrição imposta pela dependência *forte* (*início(t) > fim(t_ant)*), ficando estas tarefas dependendo da conclusão (e respectivo envio de resultados finais) de suas antecessoras diretas.

Tabela 5.3: Comportamento das tarefas durante a execução do processo com antecipação

Tarefa	*Início*	*Conclusão*	*Duração*
Discussão	0	4	4
Pesquisar trabalhos relacionados	2	8	6
Escrita da proposta	2	8	6
Realizar testes	2	7	5
Descrever trabalhos relacionados	5	9	4
Revisão da proposta	5	9	4
Descrever Testes/Resultados	5	9	4
Revisar trabalhos relacionados	7	10	3
Revisar testes	7	10	3
Conclusão/Introdução	10	14	4
Revisão Final	14	16	2
Submissão	16	16	0

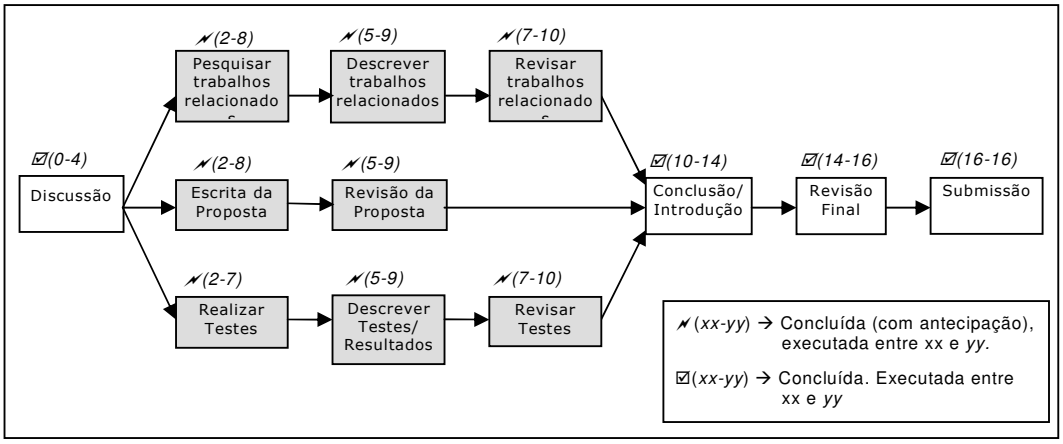

Figura 5.3: Processo executado utilizando a antecipação de tarefas

5.1.3 Comparando as duas ocorrências

A abordagem que utiliza antecipação de tarefas apresentou um tempo de execução menor, como esperado. Isto se deve ao paralelismo criado entre tarefas estritamente seqüenciais durante a execução do fluxo de trabalho. Enquanto o processo levou 22 dias para ser executado utilizando a abordagem tradicional, utilizando antecipação de algumas tarefas este tempo foi reduzido para 16 dias. A Figura 5.4 e a Figura 5.5 mostram, respectivamente, a execução do processo de maneira tradicional e utilizando antecipação, com relação à linha de tempo.

Nas figuras é possível verificar claramente o aumento do paralelismo dentro do processo. No instante 6, por exemplo, existem 6 tarefas sendo executadas concorrentemente, quando utilizando a antecipação. Isto pode trazer problemas, se for considerado que os recursos disponíveis à execução do processo são escassos e precisam ser divididos entre as tarefas. O problema de falta de recursos e ordenamento de tarefas não está no escopo do trabalho. Em nossas simulações assume-se que os recursos necessários à execução (no caso, recursos humanos) são suficientes para suprir a demanda de tarefas, havendo um agente responsável por cada tipo de tarefa prevista.

O ganho com relação ao tempo, neste caso em específico, foi de 6 dias. Além deste tipo de melhoria, existe ainda o benefício inerente à troca de resultados intermediários, que resulta no aumento na cooperação entre as tarefas. Existem, porém, alguns problemas que foram ignorados neste exemplo: o possível aumento do esforço de comunicação exigido e perda de trabalho devido a conflito de versões enquanto uma tarefa estiver sendo antecipada.

Este exemplo não foi utilizado para validar o uso ou apresentar o desempenho da utilização da antecipação de tarefas. A apresentação deste exemplo tem como objetivo apresentar uma simulação de maneira mais clara, apresentando o processo tarefa a tarefa. Este exemplo torna possível perceber o comportamento (disparo/conclusão) de cada uma das tarefas durante a execução de um processo com antecipação.

Na próxima seção serão apresentadas as simulações que foram realizadas utilizando o protótipo implementado (apresentado no capítulo 6) e o processo aqui apresentado.

Tabela 5.4: Comparativo entre a execução com antecipação e sem antecipação

Tarefa	*Tradicional*			*Com Antecipação*		
	Início	**Conclusão**	**Duração**	**Início**	**Conclusão**	**Duração**
Discussão	0	4	4	0	4	4
Pesquisar trabalhos relacionados	4	10	6	2	8	6
Escrita da proposta	4	10	6	2	8	6
Realizar testes	4	9	5	2	7	5
Escrever trabalhos relacionados	10	14	4	5	9	4
Revisão da proposta	10	12	2	5	9	4
Descrever Testes/Resultados	9	13	4	5	9	4
Revisar trabalhos relacionados	14	16	2	7	10	3
Revisar testes	13	15	2	7	10	3
Conclusão/Introdução	16	20	4	10	14	4
Revisão Final	20	22	2	14	16	2
Submissão	22	22	0	16	16	0

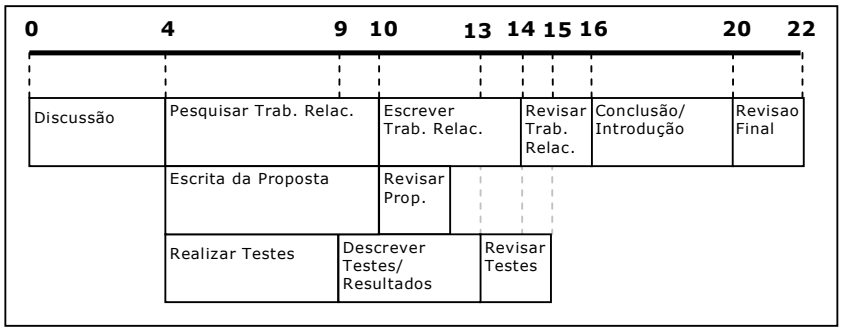

Figura 5.4: Execução do processo de maneira tradicional

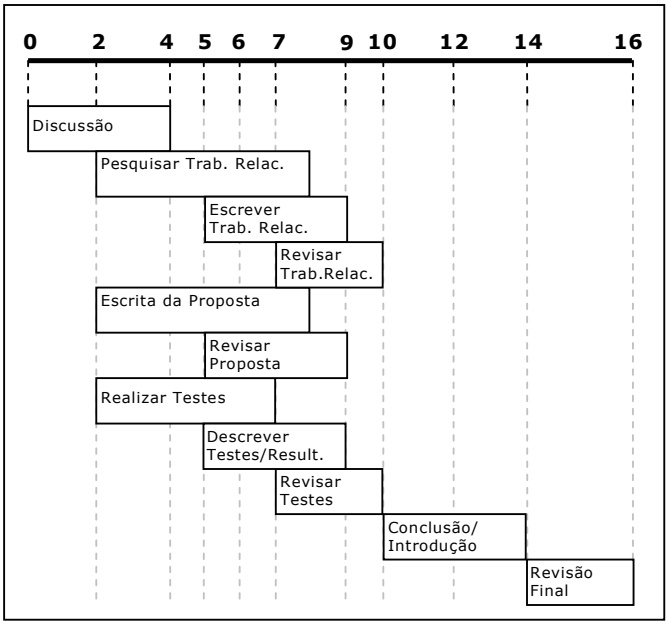

Figura 5.5: Execução do processo com antecipação de tarefas

5.2 Simulações realizadas

Nesta seção são apresentadas as simulações a que foi submetido o processo apresentado na seção anterior. O processo definido (Figura 5.1) foi submetido a simulações onde a métrica aplicada diz respeito ao tempo de execução dos processos. Este é um dos tipos de medidas fundamentais citadas por Aiello (2004), consistindo na duração de instâncias, que provê uma indicação sobre a duração de um trabalho, que é o tempo decorrido entre o início e a conclusão de uma instância.

A simulação aqui aplicada é a chamada simulação com eventos discretos (*discrete event simulation*), na qual o comportamento do sistema de *workflow* e dos atores que executam o processo são simulados (AIELLO, 2004). Nela as variáveis de um sistema mudam de valor de forma imediata, em instantes separados e específicos de tempo, chamados de eventos. Então, um evento é uma ocorrência instantânea que pode alterar o estado do sistema (TRAMONTINA, 2004).

Foram realizadas simulações utilizando as variáveis apresentadas nas opções disponíveis ao usuário no simulador implementado (seção 6.1). O uso destas variáveis visa representar mais aspectos reais na simulação, buscando a obtenção de resultados finais mais ricos. As variáveis utilizadas são explicadas a seguir:

- **Tempo de execução**: Como o próprio nome diz, esta variável armazena o tempo de execução das tarefas dentro das instâncias dos processos. Estes valores são gerados para cada instância do processo, sendo gerados aleatoriamente, seguindo uma distribuição dentre as oferecidas pelo simulador (BetaPERT, Normal e Linear).
- **Início da Antecipação**: Representa (em porcentagem) quando deverá ser o início da antecipação das tarefas. O valor desta variável representa quanto das tarefas antecessoras a uma tarefa antecipável deve estar concluído para que esta possa iniciar sua execução, representando o fornecimento de um resultado intermediário. Estes valores podem variar entre 0 (inicia-se as tarefas antecipáveis junto das antecessoras) e 1 (fluxo tradicional – dependência *forte*).
- **Esforço:** Esta variável visa simular o comportamento humano, representado o esforço que pode ser necessário à execução de uma tarefa antecipada. Este esforço compreende comunicação adicional, tempo de troca de resultados, espera por outros resultados e resolução de conflitos. Este é um valor relativo (representado em porcentagem), e pode assumir qualquer valor. Quando da simulação, é calculado o tempo absoluto (em unidades de tempo) a ser computado a cada tarefa de acordo com o valor relativo do esforço.
- **Carga de Trabalho** (*workload*): representa a freqüência com que as instâncias de processos chegam ao sistema. O valor é dado em instâncias/unidade de tempo. Se estes valores forem maiores, as instâncias chegarão mais freqüentemente ao sistema. Valores de carga de trabalho menores fazem com que as instâncias cheguem mais espaçadamente ao sistema, diminuindo a taxa de utilização dos recursos.

O primeiro cenário ao qual o processo foi aplicado foi a simulação confrontando o tempo médio de execução do processo utilizando a execução tradicional (dependência *forte*) e utilizando a antecipação (dependência *média*), prevendo o início da antecipação em diferentes pontos da execução das antecessoras, variando entre "0,1" e "1".

Em seguida, foi realizada uma bateria de simulações utilizando a variável criada para tentar representar o "esforço adicional" exigido pela utilização da antecipação. O intuito desta simulação é criar um padrão de comportamento na relação entre esforço e início da antecipação, a fim de prever quais as situações em que a antecipação é vantajosa. Confrontando os dados aqui obtidos com casos reais, pode-se obter as condições em que se deve, ou não se deve permitir a antecipação. Os valores de esforço variaram entre "0" e "1" e os valores de início da antecipação variaram entre "0,1" e "1".

O processo de criação de um artigo científico foi submetido a estas simulações de duas maneiras diferentes. Numa primeira os dados relativos ao tempo de execução das tarefas foram mantidos como apresentado na seção 5.2. Já na segunda, os dados referentes ao tempo de execução foram mapeados de dias para horas, a fim de obter resultados com uma maior precisão, já que se está trabalhando com números inteiros. Para conversão foi tomado que 1 dia equivale a 8 horas.

Por último, a execução do processo foi simulada com uma carga de trabalho (*workload*) variando entre "0,1" e "1" instância disparada ao dia. Foram realizadas simulações sem considerar esforço adicional, para valores de início da antecipação entre 0,1 e 1.

Tabela 5.5: Duração Mínima, Máxima e Média para as tarefas – em dias e em horas

Tarefa	***Duração Mínima***	***Duração Máxima***	***Duração Média***
Discussão	4 (32)	4 (32)	4 (32)
Pesquisar trabalhos relacionados	4 (32)	8 (64)	6 (48)
Escrita da proposta	4 (32)	8 (64)	6 (48)
Realizar testes	3 (24)	7 (56)	5 (40)
Descrever trabalhos relacionados	3 (24)	5 (40)	4 (32)
Revisão da proposta	1 (8)	3 (24)	2 (16)
Descrever Testes/Resultados	3 (24)	5 (40)	4 (32)
Revisar trabalhos relacionados	1 (8)	3 (24)	2 (16)
Revisar testes	1 (8)	3 (24)	2 (16)
Conclusão/Introdução	3 (24)	5 (40)	4 (32)
Revisão Final	1 (4)	3 (24)	2 (16)
Submissão	0 (0)	0 (0)	0 (0)

5.2.1 Resultados

Cada cenário de simulação contou com 500 instâncias passando pelo sistema. Os valores finais de cada uma das variantes do processo foram colhidos e seus valores médios foram calculados. Para todas as simulações os valores de tempo de execução das tarefas foram gerados de acordo com uma distribuição BetaPERT. Os valores de esforço quando gerados aleatórios, seguiram uma distribuição linear, assim como os valores de início da antecipação. Quando se utilizou *carga de*

trabalho, a estratégia de ordenamento das tarefas utilizadas foi a FCFS (*First to Come, First to Serve*). Os valores de duração máxima, mínima e média de cada tarefa foram apresentados na seção anterior, e estão reproduzidos na Tabela 5.5. Os valores apresentados entre parênteses representam o mapeamento do tempo para horas. A seguir são apresentados os resultados obtidos em cada um dos cenários de simulação propostos.

5.2.1.1 Tempo médio de execução dos processos

Os resultados relativos ao tempo médio de execução do processo variando-se o início da antecipação são apresentados na Tabela 5.6. Neste primeiro cenário não é considerada a existência de qualquer esforço extra oriundo da antecipação.

Como era esperado, a utilização da antecipação de tarefas fez com que o tempo de execução dos processos caísse com relação à execução baseada na dependência forte (*fim-início*). Os números mostram que, quanto antes for iniciada a antecipação das tarefas, melhor o desempenho apresentado na execução do processo.

Se forem comparadas as execuções segundo a abordagem tradicional e a abordagem com antecipação iniciando-se em 10% da conclusão das tarefas antecessoras, existe um ganho de tempo superior a 30%. Para o valor intermediário de início da antecipação (50%) este ganho é de aproximadamente 20%. Quando o início da antecipação é gerado aleatoriamente para cada uma das tarefas (variando entre 10% e 100%), obtém-se um ganho muito próximo ao obtido com o início em 50% (cerca de 20%).

Tabela 5.6: Tempo médio de execução dos processos variando o início da antecipação

Início da Antecipação	Tempo Médio
10%	15,69
20%	15,83
30%	16,72
40%	17,25
50%	19,21
60%	19,90
70%	20,32
80%	22,43
90%	23,33
Aleatório	19,32
Execução Tradicional	23,66

Observando-se o gráfico gerado pelos valores da tabela (Figura 5.6) pode-se perceber a existência de alguns “patamares” seguidos de um aumento nos valores de tempo de execução. Isso ocorre, pois os valores utilizados para tempo de execução são números inteiros e o tempo de execução das tarefas são valores geralmente pequenos (entre 0 e 8). Tomando como exemplo uma tarefa com duração de 5 dias e considerando que esta fornece um resultado intermediário com 10% de sua execução, este resultado será fornecido no dia 1. Se o resultado for fornecido com 20% de sua execução, isto também ocorrerá no dia 1.

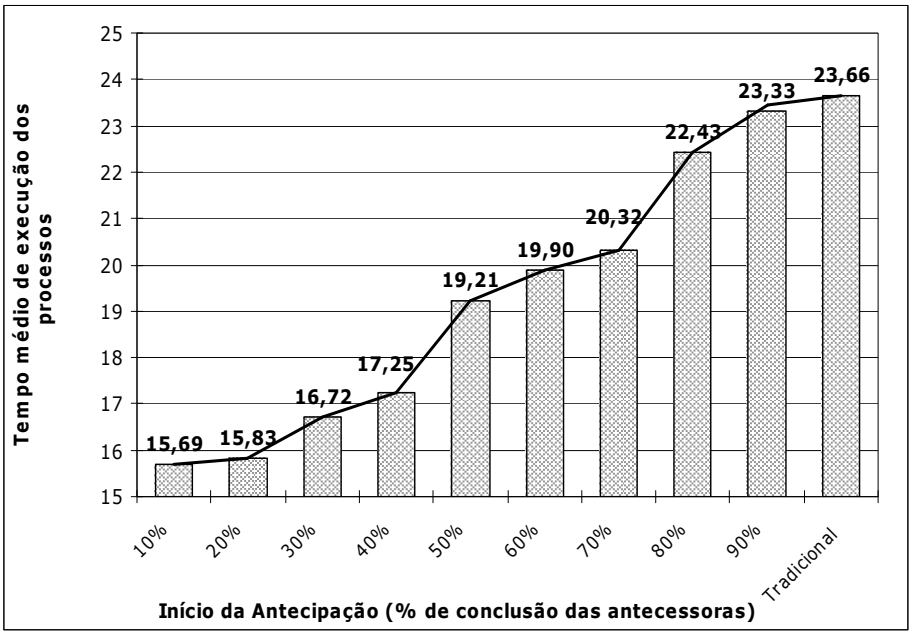

Figura 5.6: Tempo Médio de execução dos processos

Para amenizar este tipo de problema e tentar obter resultados mais claros, os valores de tempo de execução das tarefas foram mapeados para horas. Após a realização das mesmas simulações, chegou-se aos resultados da Tabela 5.7.

Tabela 5.7: Tempo médio de execução das tarefas (em horas) variando o início da antecipação

Início da Antecipação	Tempo Médio
10%	119,89
20%	123,32
30%	126,87
40%	132,68
50%	140,52
60%	149,34
70%	156,28
80%	164,46
90%	171,83
Aleatório	147,38
Execução Tradicional	178,67

A simulação utilizando valores maiores (horas ao invés de dias) apresentou resultados mais claros para este cenário, onde se desejava obter uma relação entre início da antecipação e tempo de execução. Observando-se o gráfico percebe-se que o tempo médio de execução dos processos cresce linearmente com o aumento do início da antecipação das tarefas.

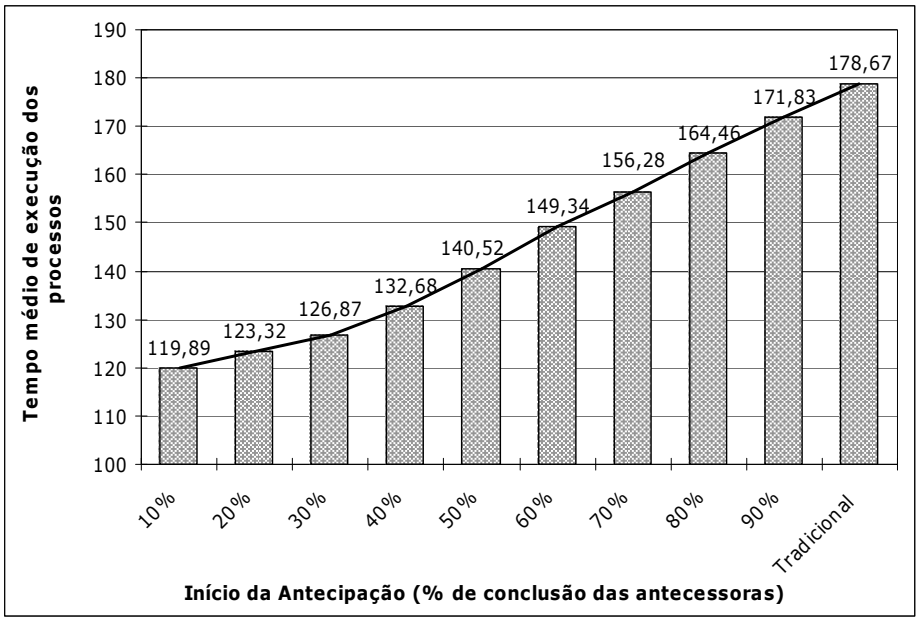

Figura 5.7: Tempo Médio de execução dos processos com valores mapeados para horas

Com relação aos ganhos, os resultados são similares aos calculados quando da utilização de dias. Comparando-se o início antecipado aos 10% das anteriores obteve-se um ganho de 33% com relação à abordagem tradicional. Com a variável de início da antecipação definida como aleatória, o desempenho apresentou-se 18% melhor.

As simulações realizadas levando em conta apenas o início da antecipação mostraram que o tempo médio de execução dos processos cai quando é prevista a antecipação. É válido lembrar que este é o caso ideal, onde não são simuladas possíveis perdas relacionadas ao esforço de comunicação exigido para troca de resultados e resolução de conflitos. O próximo cenário utiliza uma variável que simula tal esforço.

5.2.1.2 Tempo médio de execução dos processos com esforço adicional

O segundo cenário de simulação ao qual o processo foi submetido utilizou, além da variável de início da antecipação, a variável que simula o esforço oriundo da antecipação de tarefas. Os resultados obtidos com a variação destes valores durante a simulação são apresentados na

Tabela 5.8 e na Figura 5.8.

Os resultados aqui obtidos visam produzir um padrão de comportamento relacionando esforço necessário para antecipar as tarefas e quando se dá o início da antecipação das tarefas. A partir dos dados e do gráfico é possível verificar para quais valores de esforço e início da antecipação é vantajoso utilizar a antecipação (em termos de tempo de execução). Com estes resultados confrontados com histórico de casos reais de execução de processos pode-se prever quando seria adequado liberar ou bloquear a antecipação de uma tarefa, por exemplo.

Tabela 5.8: Tempo médio de execução das tarefas variando o início da antecipação e o esforço adicional

Início \ Esforço	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Aleatório
10%	15,66	16,60	17,75	18,28	18,90	19,32	19,96	20,85	21,16	22,33	22,36	19,57
20%	15,76	16,83	17,83	18,36	18,96	19,43	20,19	21,07	21,25	22,42	22,54	20,02
30%	16,68	18,13	18,70	19,61	20,00	20,71	21,53	22,15	22,76	23,61	23,76	21,33
40%	17,28	18,77	19,27	20,11	20,65	21,26	22,06	22,76	23,66	24,22	24,16	22,09
50%	19,17	20,80	21,42	22,23	22,83	23,37	24,58	25,32	26,09	27,24	27,32	24,21
60%	19,83	21,78	22,27	23,14	24,14	24,26	25,69	26,85	27,06	28,58	28,38	25,31
70%	20,27	22,79	23,42	24,41	25,21	25,72	27,05	28,12	29,07	30,26	30,33	26,59
80%	22,69	24,53	25,25	26,29	27,51	28,33	29,82	31,08	31,71	32,66	32,47	28,80
90%	23,44	25,93	27,11	28,31	29,31	30,24	31,72	32,56	33,87	34,60	34,62	30,75
Aleatório	19,23	21,14	22,02	23,03	23,93	24,17	24,81	26,15	27,33	27,85	28,31	25,03
Execução Tradicional	23,67	23,67	23,67	23,67	23,67	23,67	23,67	23,67	23,67	23,67	23,67	23,67

Segundo os números, a antecipação traz melhores resultados que a execução tradicional quando as tarefas são antecipadas no início da execução das antecessoras (10% e 20%). As vantagens aparecem mesmo que o esforço necessário à antecipação seja de 100%, isto é, mesmo que o tempo de execução seja duplicado.

Para o início acontecendo entre 30% e 40% da execução das antecessoras, as vantagens aparecem para valores de esforço inferiores a 80%. As vantagens para início com 50% das antecessoras completas aparecem quando o esforço é inferior a 50%. Valores de início da antecipação acima de 60% apresentam desempenho mais baixo que a execução tradicional quando o esforço ultrapassa 20%.

Fica explícito que o início da antecipação tardio não traz ganhos representativos ao processo. Iniciando-se a antecipação após 80% da execução das antecessoras os ganhos aparecem apenas quando não se tem esforço, não sendo vantajosa em nenhum dos outros casos. Porém, quando o início da antecipação é tardio, os resultados fornecidos são praticamente resultados finais, e o esforço adicional não deve apresentar-se alto.

Com valores aleatórios de início da antecipação as vantagens aparecem para valores menores de 50% de esforço adicional. Já para valores aleatórios de esforço adicional as vantagens aparecem até com início da antecipação igual a 50%. Quando ambos os valores são obtidos de maneira aleatória, os resultados apresentaram-se satisfatórios, ficando a média praticamente igual à média obtida com a execução tradicional.

Através destes valores, podemos criar intervalos onde a antecipação é vantajosa. Supondo que análise histórica de um sistema mostre que o esforço médio inserido pela antecipação é de 40%. Observam-se os valores que ficam abaixo do tempo médio de execução utilizando a abordagem tradicional quando o esforço introduzido é de 40%. Neste caso, a antecipação mostra-se vantajosa seu início dá-se até com 50% da execução das antecessoras concluídas. O gerente do sistema pode então optar por não disponibilizar a antecipação quando a execução das antecessoras superarem estes 50%.

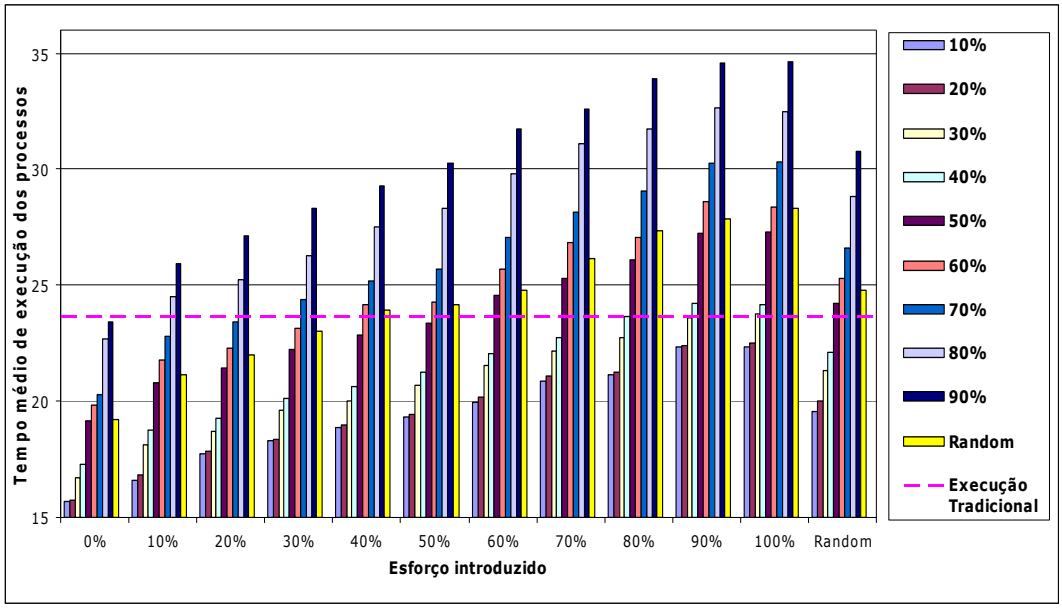

Figura 5.8: Tempo Médio de execução dos processos levando em conta o início da antecipação e o esforço adicional

Mais uma vez as simulações foram repetidas para valores em horas, ao invés de dias. Os resultados obtidos são apresentados na Tabela 5.9, e, mais uma vez, apresentaram um padrão de comportamento muito mais claro. Apesar disso, a precisão dos resultados não afetou muito as conclusões relativas aos valores.

A principal alteração nos resultados diz respeito à antecipação em 30%, que apresentou melhor desempenho que a execução tradicional em todos os casos. Valores de início da antecipação iguais a 40%, 50% e 60% também se mostraram melhores, apresentando vantagem com relação à execução tradicional para valores de esforço inferiores a 90%, 70% e 50% respectivamente.

Tabela 5.9: Tempo médio de execução das tarefas (em horas) variando o início da antecipação e o esforço adicional

Início \ Esforço	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Aleatório
10%	119,89	124,39	130,57	135,64	140,53	144,59	148,66	155,46	159,16	165,10	169,15	148,51
20%	123,32	129,08	133,35	138,30	143,71	147,91	152,30	158,20	163,21	168,09	173,62	153,46
30%	126,87	133,52	138,03	142,84	148,66	155,64	158,14	164,41	168,41	173,51	178,27	158,12
40%	132,68	138,96	144,53	150,04	155,22	159,49	165,16	170,98	175,81	180,91	186,94	164,39
50%	140,52	147,92	153,54	159,37	164,50	169,78	175,84	181,56	187,01	193,33	198,69	173,70
60%	149,34	155,72	161,91	168,59	174,51	180,04	186,61	193,75	199,09	205,29	210,97	183,26
70%	156,28	163,44	170,61	178,14	184,36	190,10	196,48	204,40	210,75	216,78	224,06	194,44
80%	164,46	172,08	179,63	186,13	192,86	199,50	207,23	215,86	221,85	230,54	236,68	204,09
90%	171,83	181,21	188,67	195,90	203,97	211,04	219,52	225,84	233,34	240,86	247,33	215,29
Aleatório	147,38	153,27	158,86	166,02	172,11	176,74	183,06	189,04	197,40	203,00	207,73	179,20
Execução Tradicional	178,67	178,67	178,67	178,67	178,67	178,67	178,67	178,67	178,67	178,67	178,67	178,67

Quando aplicados valores aleatórios às variáveis, os resultado são muito similares àqueles obtidos com duração em dias. Porém, a execução com valores aleatórios de esforço e início de antecipação de 50% apresentou ganhos significativos em relação à execução tradicional. Quando as duas variáveis foram tomadas aleatoriamente, houveram ganhos, porém menores. O gráfico com estes resultados é apresentado na Figura 5.9.

Após observar a simulação com esforço para o cenário utilizando dias e horas a conclusão que se chega é que, quanto antes se inicia a antecipação, melhores são os resultados. Porém, esta pode gerar um pior desempenho quando iniciada após 70% da execução das tarefas antecessoras, principalmente se houver muito esforço adicional relacionado à troca de resultados intermediários.

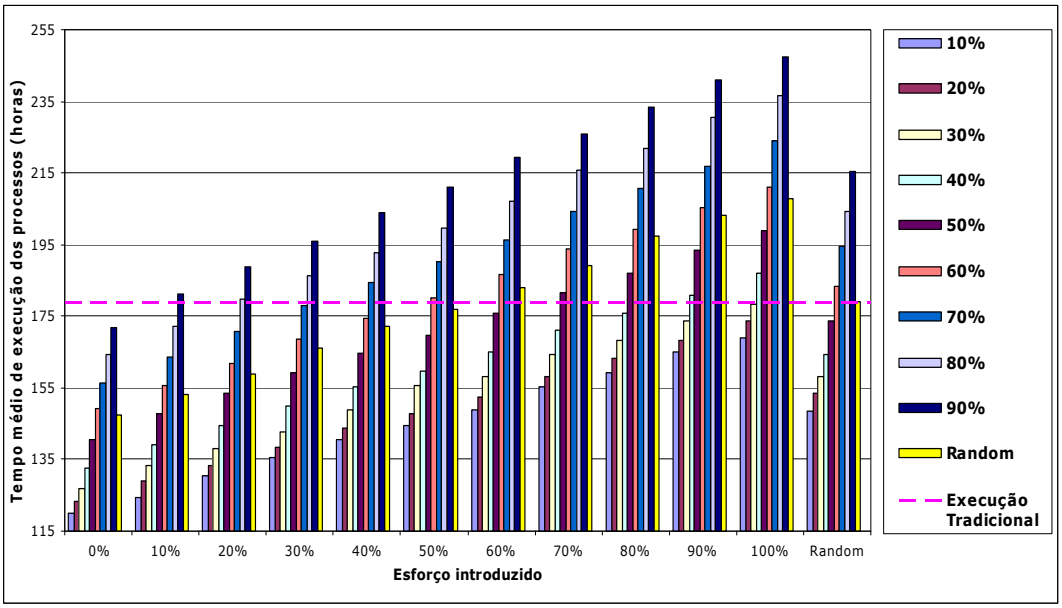

Figura 5.9: Tempo Médio de execução dos processos levando em conta o início da antecipação e o esforço adicional com valores mapeados para horas

5.2.1.3 Tempo médio de execução dos processos com carga de trabalho

O último cenário de simulações combinou carga de trabalho (*workload*) e início da antecipação. Os resultados desta simulação são apresentados na Tabela 5.10, e, graficamente, na Figura 5.10.

Tabela 5.10: Tempo médio de execução das tarefas variando o início da antecipação com *workload*

Workload / Início	0,100	0,125	0,200	0,250	0,500	1,000
10%	15,57	15,58	66,03	115,63	214,63	264,13
20%	15,74	15,75	66,23	115,79	214,79	264,29
30%	16,75	16,75	67,47	116,94	215,94	265,44
40%	17,19	17,19	67,98	117,39	216,39	265,89
50%	19,00	18,92	69,69	119,12	218,12	267,62
60%	19,65	19,65	70,29	119,74	218,74	268,24
70%	20,33	20,33	70,89	120,34	219,34	268,84
80%	22,16	22,19	72,78	122,23	221,23	270,73
90%	23,47	23,49	73,99	123,49	222,49	271,99
Execução Tradicional	23,64	23,68	73,04	123,49	222,49	269,10

Conforme os valores de carga de trabalho aumentavam os valores de tempo de execução cresceram. Porém, se for analisada a diferença existente entre os valores com antecipação e o valor da execução tradicional, não ocorreram mudanças.

O que se esperava com este cenário de simulação era verificar se a antecipação de tarefas poderia melhorar o tempo médio de execução com o aumento do *workload*. Esta hipótese não se confirmou. Observa-se que, iniciando-se a antecipação com 10% da execução das antecessoras, o ganho varia entre 7 e 8 dias no tempo médio, para todos os valores de carga de trabalho. Para 50% essa diferença está em 4 dias. O mesmo ocorre para os outros valores de início de antecipação.

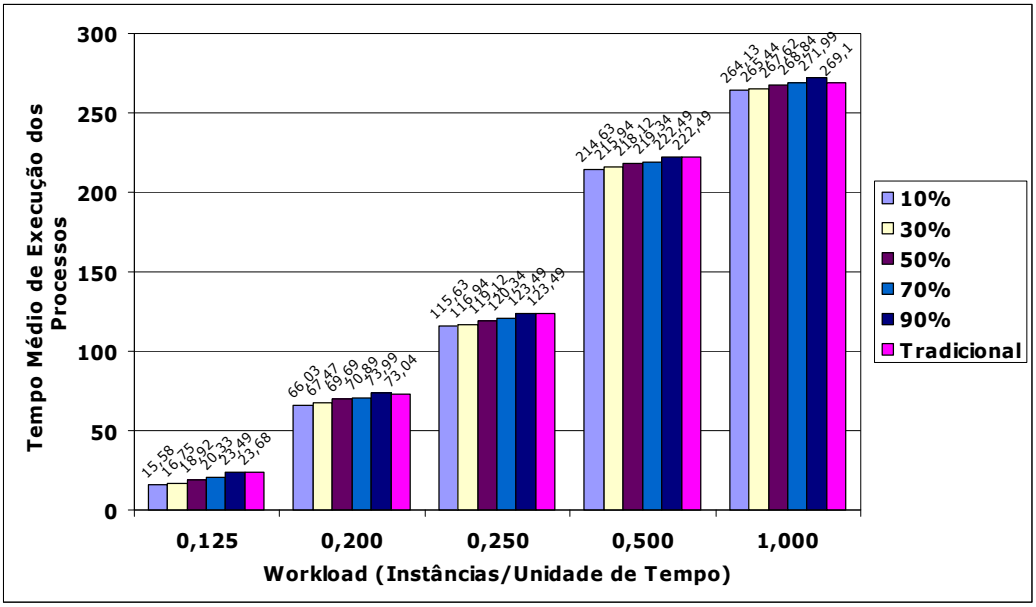

Figura 5.10: Tempo Médio de execução dos processos com carga de trabalho

Os valores de tempo médio de execução crescem linearmente, não havendo variação considerável com o aumento da carga de trabalho. Apesar de não crescerem, os ganhos da antecipação com relação ao tempo médio continuam existindo.

5.3 Considerações Finais

Os resultados apresentados nesta seção mostram a possibilidade de melhoria de desempenho com relação ao tempo de execução dos processos através da utilização da antecipação de tarefas. De maneira geral pode-se dizer que os resultados são bons, mostrando a vantagem que se pode tirar da antecipação.

Ficou demonstrado que, para a antecipação ser mais eficiente, é necessário que esta seja iniciada antes da metade da execução das tarefas antecessoras. Mesmo se houver esforço adicional a antecipação mostra-se vantajosa para alguns casos.

Antecipando-se tarefas próximo ao final de suas antecessoras pode trazer perdas, se o esforço necessário para tal antecipação existir.

As simulações realizadas variando o esforço e o início da antecipação podem ser utilizadas para limitar o período em que é permitida a antecipação das tarefas. Estes resultados podem ser utilizados junto de uma análise histórica de esforço devido à antecipação, gerando um intervalo real em que a antecipação é vantajosa.

Os resultados aqui apresentados dizem respeito apenas a um dos pontos em que a antecipação pode trazer vantagens: o tempo de execução. As vantagens de utilizar a antecipação extrapolam esta aqui apresentada. Pode ser que, apesar de resultados em termos de tempo não favoráveis, a antecipação seja utilizada para aumentar a cooperação dentro do grupo e a qualidade do produto final.

6 IMPLEMENTAÇÕES

Para validar e prover meios de utilizar a antecipação de tarefas foram propostas algumas implementações. A primeira delas é um simulador que permite a execução de processos levando em conta a antecipação. Este simulador foi utilizado para obtenção dos resultados apresentados no capítulo 5.

Foram implementadas junto ao Amaya *Workflow* (TELECKEN, 2004) as funcionalidades de antecipação na fase de definição de processos. Estas funcionalidades permitem a definição das tarefas antecipáveis, possibilitando sua exportação para XPDL. Por último, foi proposta a modificação da máquina de *workflow* implementada por França (2004) no projeto CEMT (LIMA *et al.*, 2001) para suportar a antecipação de tarefas.

6.1 Simulador

A implementação de um simulador específico para este trabalho está relacionada às necessidades específicas dos testes pretendidos. A maior motivação para tal implementação é a inexistência de meios simples de simular a antecipação de tarefas em simuladores de processos tradicionais. Tal tarefa seria muito complexa e demandaria muito tempo, pois seria necessário utilizar a antecipação gerenciada *a priori*, sendo necessário modificar a granularidade e o tempo de execução de cada tarefa manualmente a cada simulação.

O protótipo de simulador implementado possui a arquitetura apresentada pela Figura 6.1. Qualquer definição de processo representada em um arquivo XPDL pode ser importada pela ferramenta, podendo assim ser simulada. Portanto, qualquer ferramenta de definição de processos que respeite as recomendações da WfMC poderá ter seus processos simulados por tal protótipo. Ao ser importado, o processo representado em XPDL é transformado em uma *query* SQL (*Structured Query Language*), que é executada por um sistema gerenciador de banco de dados, persistindo os dados relevantes à simulação. Estes dados são então manipulados pelo motor de execução, que simulará desde o disparo das tarefas até a execução das mesmas. A resposta obtida é dada em termos do tempo de execução total dos processos.

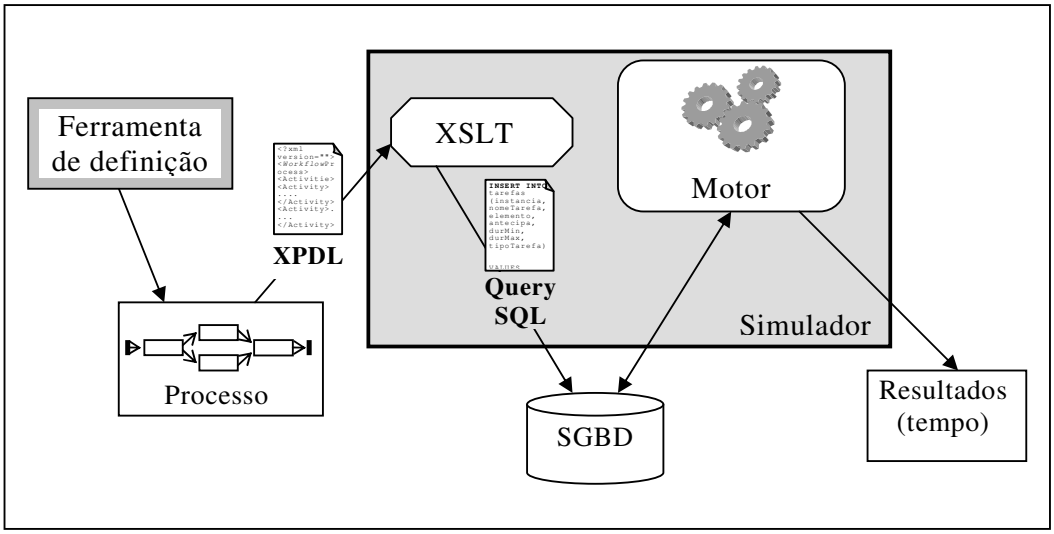

Figura 6.1: Arquitetura do protótipo de simulação implementado

A seguir são especificados cada um dos elementos apresentados na arquitetura apresentada (Figura 6.1).

6.1.1 Ferramenta de definição

Qualquer ferramenta de definição que possua a funcionalidade de exportação para o formato XPDL pode ser utilizada para modelar o processo a ser simulado. A única exigência é que seja um formato XPDL válido, de acordo com a definição da WfMC.

Para o presente trabalho foi utilizada a ferramenta de definição Amaya *Workflow* (TELECKEN, 2004). Esta ferramenta foi estendida para suportar os atributos relativos à antecipação de tarefas como é descrito na seção 6.2. Outra extensão necessária foi a inclusão de dois novos atributos que identificassem os tempos de duração mínimo e máximo de cada uma das tarefas. Assim, como o atributo "*wAntecipa*" (seção 6.2), estes também foram definidos como *Extended Attributes*, previstos na linguagem XPDL.

Na ferramenta *Amaya Workflow*, estes atributos (*wDurMin* e *wDurMax*) aparecem como um novo item no menu *Attributes*, quando um elemento tarefa está selecionado. Clicando sobre eles, uma caixa de texto aparecerá, onde deve ser preenchido um número inteiro. O preenchimento dos dois atributos representará o possível intervalo de duração da tarefa em questão. Estes elementos são apresentados junto do atributo *wAntecipa* na Figura 6.7.

Quando exportado para XPDL, estes atributos aparecerão como *ExtendedAttributes* da tarefa à qual estão relacionados. O aninhamento de tais atributos dentro de uma tarefa é apresentado no trecho do processo representado em XPDL apresentado a seguir:

```
...
  <Activity Name= "Tarefa1" Id= "T1" >
      <Implementation>
          <Tool Id="checkData" Type="APPLICATION">
              <Participant>
                  ...
              </Participant>
              <ActualParameters>
                  ...
                  ...
              </ActualParameters>
          </Tool>
      </Implementation>
      <ExtendedAttributes>
          <ExtendedAttribute Name="antecipa" value="N"/>
          <ExtendedAttribute Name="durMin" value="8"/>
          <ExtendedAttribute Name="durMax" value="15"/>
      </ExtendedAttributes>
  </Activity>
```

Figura 6.2: Trecho de uma definição de processo em XPDL explicitando os atributos estendidos durMin e durMax

6.1.2 XSLT

A fim de importar o processo gerado pela ferramenta de definição de processos, optou-se pela implementação de uma "folha de estilos" XSLT (eXtensible Stylesheet Language Transformations). Esta opção foi feita uma vez que a linguagem de representação de processos XPDL é baseada em XML, e a linguagem de transformação XSLT foi desenvolvida justamente para transformar um documento XML em outro documento com outro formato.

No caso do presente trabalho, foi necessário criar uma folha de estilos capaz de transformar alguns dos elementos previstos na linguagem XPDL para um *script* SQL. Estes elementos são basicamente informações básicas sobre o processo, as tarefas, relacionamentos entre as tarefas (conectores) e informações relativas à simulação. A Tabela 6.1, mostra alguns dos elementos que foram importados para possibilitar a simulação dos processos.

Com relação à tabela apresentada, são necessárias algumas explicações. O conjunto reduzido de informações importadas ao simulador é um conjunto que permite o funcionamento do mesmo. Algumas outras informações como *deadline*, data de início, data de término, prioridade das tarefas e informações dos agentes participantes, também são importantes para simulação, porém, não são utilizadas nas simulações que são aqui apresentadas.

Com relação ao tipo da tarefa, são previstos, basicamente, 5 tipos de tarefas quando da importação: *split* (ou *fork*), *join*, tarefa comum, *begin* e *end*. As tarefas dos tipos *split* e *join* são explicitamente definidas no XPDL, enquanto as outras não. Quando da importação, estes dois tipos de tarefas são identificados, sendo todas as outras importadas como tarefas comuns. A distinção das tarefas do tipo *begin* (inicial) e *end* (final) é realizada quando da primeira instanciação do processo, através da análise de seus conectores de entrada e saída.

Tabela 6.1: Informações importadas dos arquivos XPDL para permitir a simulação

Informações sobre tarefas e processos (XPDL)	
Id do Processo	<*Workflow*Processes/*Workflow*Process/@Id>
Nome da Tarefa	<//Activities/Activity/@name>
Id da Tarefa	<//Activities/Activity/@id>
Tipo da Tarefa	<//TransitionRestrictions/TransitionRestriction/Join>
	<//TransitionRestrictions/TransitionRestriction/Split>
Antecipável (S/N)	<//ExtendedAttributes/ExtendedAttribute/@Name=antecipa>
Duração Mínima	<//ExtendedAttributes/ExtendedAttribute/@Name=durMin>
Duração Máxima	<//ExtendedAttributes/ExtendedAttribute/@Name=durMax>
Conectores (De → Para)	<//Transitions/Transition/@From>
	<//Transitions/Transition/@To>

6.1.3 SGBD

O resultado da transformação do processo representado em XPDL, como dito, é um *script* SQL, que é executado em um SGBD. Neste trabalho é utilizado o SGBDR MySQL (2004), devido á facilidade de uso, ampla utilização e por tratar-se de um software livre, seguindo a proposta inicial dos trabalhos realizados no contexto do projeto CEMT (LIMA *et al.,* 2001).

O armazenamento das informações importadas resume-se a quatro tabelas. Uma delas armazena as informações sobre os processos. A segunda apresenta as informações sobre as tarefas. Uma outra tabela, apresenta informações relativas aos participantes. E, a última, traz os relacionamentos entre as tarefas (conectores). Tais tabelas e seus relacionamentos podem ser vistas na Figura 6.3.

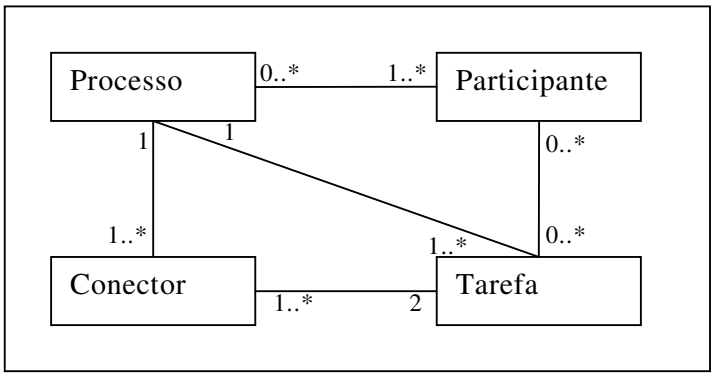

Figura 6.3: Estrutura do banco de dados que permitiu a importação dos dados

A tabela Tarefa armazena a maior parte das informações importantes à simulação em si. Dentro dela são encontrados os campos relativos à antecipação (antecipável/não Antecipável), duração das tarefas, tipo da tarefa. Outra tabela importante para o simulador aqui apresentado é a tabela de conectores, que apresenta todos os conectores de uma tarefa. O relacionamento existente entre Tarefa e Conector é do tipo 2 para muitos, uma vez que cada conector relaciona exatamente duas tarefas, e uma tarefa pode possuir vários conectores.

6.1.4 Motor

É o elemento fundamental do simulador. Foi implementado utilizando a linguagem PHP, por ser software de distribuição livre e ser uma tecnologia *web*, podendo ser utilizado através de um *browser*. Para a implementação de algumas funções matemáticas foi utilizada a biblioteca SpecialMath (VAN KOOTEN; MEAGHER, 2004).

Este motor é quem realmente simula a execução dos processos de *workflow*, e foi construído com base no funcionamento previsto pela WfMC. Ele funciona de acordo com o diagrama de estados das tarefas proposto no capítulo 4, executando a troca de estados segundo as pré-condições das tarefas.

Para o protótipo implementado estão previstas a execução tradicional de processos (dependência *forte*) e a execução com antecipação. Esta última, utilizando *dependência média*.

Este protótipo possui apenas um conjunto de funcionalidades que são necessárias ao presente trabalho. Decidiu-se fazer algumas considerações que simplificassem o processo de simulação:

- Os valores internos ao simulador são dados em números inteiros. Isto torna o simulador independente do tempo real, e, portanto questões como velocidade dos computadores não influenciam os resultados finais obtidos (TRAMONTINA, 2004).
- Todas as tarefas antecipáveis obedecem à *dependência média* de fluxo de controle. Devido à dependência com relação à conclusão das tarefas, *fim(t)* somente é possível um dia após *fim(t_ant)* se concretizar.
- Uma tarefa libera um resultado intermediário sempre que atingir um determinado ponto de sua execução. Isto é, quando todas as antecessoras de uma tarefa antecipável atingirem uma determinada porcentagem de sua execução, ela poderá iniciar a antecipação.
- Para a execução de uma instância de processo os agentes são suficientes, não havendo necessidade de haver uma fila de tarefas disponíveis para execução (ou antecipação). Isto é, o número de agentes é suficiente para suprir as necessidades de uma instância de processo.
- Uma tarefa inicia sua execução imediatamente após o término de suas antecessoras diretas. Isto é, assim que uma tarefa entra no estado *executável*, ela é solicitada para execução e passa a estar *executando*.

- O mesmo acontece com tarefas antecipáveis, que iniciam sua antecipação assim que suas pré-condições forem satisfeitas. Quando a tarefa passa para o estado *pronta.paraAntecipar* esta passa imediatamente a estar *antecipando*.
- No momento apenas são importados e simulados processos com um nível, isto é, sub-processos não são importados.

Para a execução das simulações o usuário é responsável por fornecer alguns parâmetros ao sistema. Para os fins específicos deste trabalho, são disponibilizadas opções com relação ao tipo de simulação (com ou sem antecipação), quantidade de simulações consecutivas, início da antecipação e esforço de comunicação a ser inserido. Também estão previstas opções relativas à priorização de execução de tarefas. Estas opções são apresentadas na próxima seção.

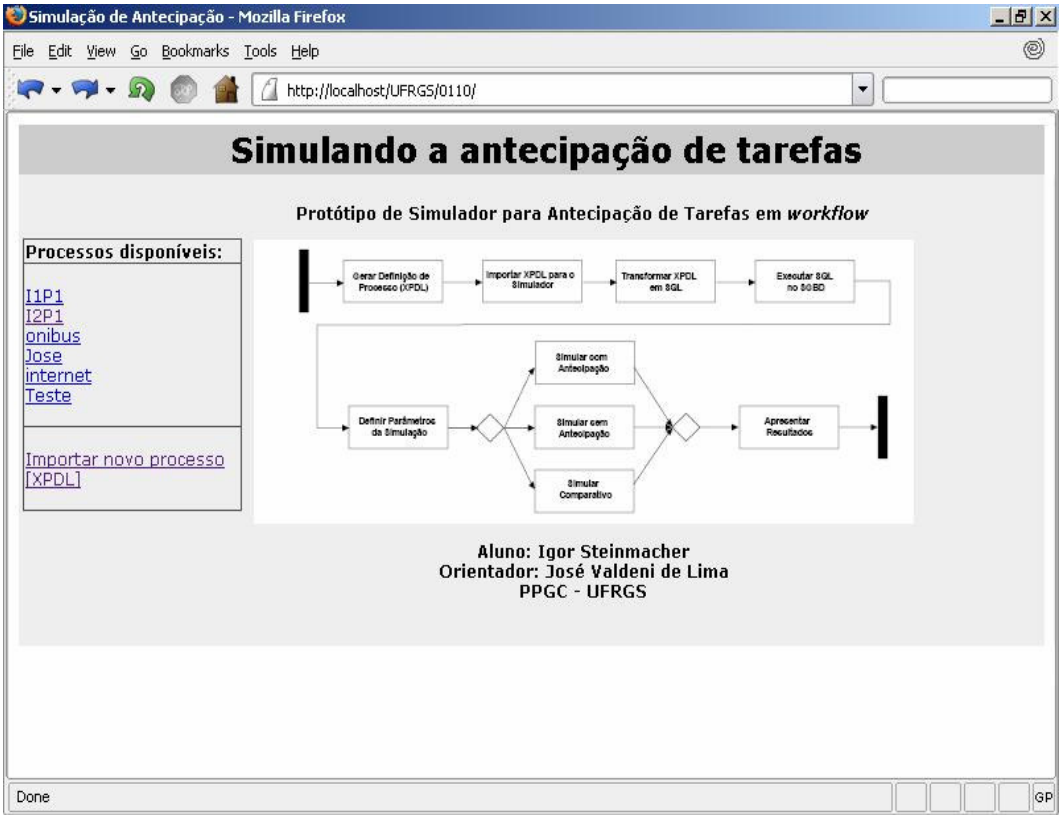

Figura 6.4: Interface inicial do protótipo do simulador implementado

6.1.5 Interface com o Usuário

A interface com o usuário se dá através de um navegador *web* qualquer, uma vez que todo o protótipo é implementado na linguagem PHP. Através desta interface é possível importar os processos, definir os parâmetros de simulação, e executar a simulação. Depois de executada a simulação é apresentado o resultado ao usuário, em termos do tempo de execução do processo.

A Figura 6.4 apresenta a interface inicial do simulador implementado. Basicamente esta interface está dividida em duas partes: um menu no lado

esquerdo da tela e uma área principal. No menu à esquerda encontram-se os processos que estão disponíveis no banco de dados. Neste mesmo menu existe a opção de importação de um novo processo, representado em XPDL. Na área principal é apresentada a página de abertura do protótipo, com a ilustração de um *workflow* do funcionamento do próprio simulador.

Através do menu encontrado ao lado esquerdo, o usuário pode clicar em um dos processos mostrados para iniciar a simulação do mesmo. Ao executar esta operação, será mostrada uma série de opções relativas à simulação. A Figura 6.5 mostra as opções apresentadas ao usuário.

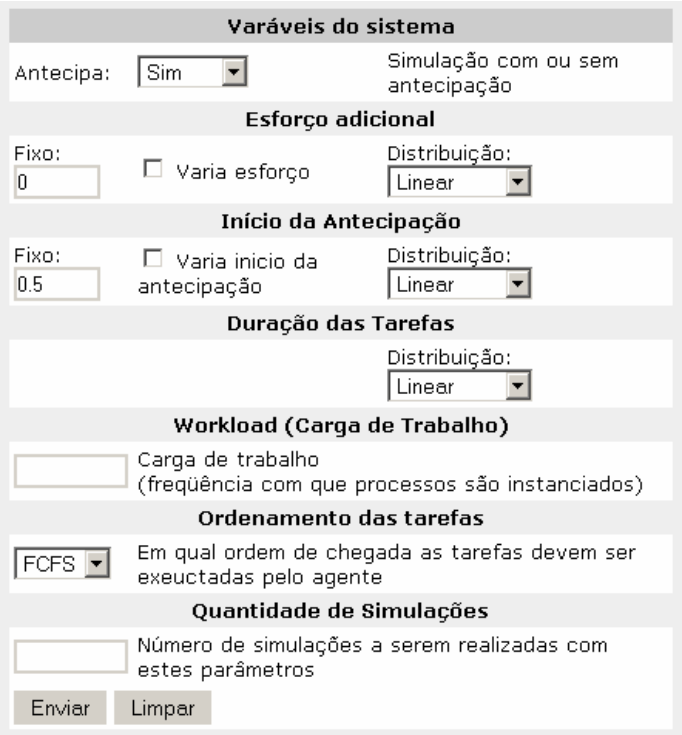

Figura 6.5: Opções de simulação previstas no protótipo implementado

A primeira opção apresentada diz respeito ao uso da antecipação. Através dela o usuário pode escolher se a simulação levará em conta a antecipação de tarefas ou não. Também é possível optar pela execução de ambas maneiras, isto é, simular o processo com antecipação e sem antecipação, de maneira a comparar as execuções mais facilmente.

As opções relacionadas ao "Esforço Adicional" podem ser utilizadas quando se deseja simular a possibilidade de aumento no tempo de execução das tarefas devido ao aumento no esforço. Este atua nas tarefas que estão participando da antecipação, tentando simular o esforço de comunicação, troca de resultados e cooperação inerentes a estas tarefas. Se for marcada a opção "varia Esforço" esta opção é descartada e o esforço é gerado aleatoriamente durante a simulação. Caso não seja marcada a opção, o sistema levará em consideração o valor de esforço fixo da caixa de texto.

As opções de “Início da Antecipação” são utilizadas para informar ao sistema o valor da variável que indica o momento do disparo das tarefas antecipáveis com relação ao tempo de execução de suas antecessoras. O Início Fixo é utilizado quando se deseja manter constante o momento do início da antecipação das tarefas com relação às suas antecessoras. Este campo deve ser preenchido com valores entre “0” e “1” (0% e 100%), representando depois de quanto tempo de execução de suas antecessoras uma tarefa antecipável poderá iniciar sua execução. Existe também a opção “Varia início da antecipação”, através da qual o usuário pode solicitar que o sistema decida de maneira aleatória este valor para cada instância de tarefa.

Para todas as opções que consideram a variação de algum dos parâmetros (esforço, início da antecipação e duração das tarefas) é oferecida ao usuário a possibilidade de escolha do tipo de distribuição a ser utilizada. No presente protótipo existem três distribuições previstas: linear, normal e betaPERT. A distribuição linear gera valores igualmente distribuídos tendo como entrada os valores limites do intervalo (máximo e mínimo). A distribuição normal é gerada aqui através dos valores de máximo e mínimo, sendo a média central e o desvio padrão obtido através do modelo de distribuição triangular. A distribuição betaPERT tem como entrada os valores de mínimo e máximo e a média sendo central.

As duas próximas opções dizem respeito ao ordenamento das tarefas. Através da primeira é possível definir a carga de trabalho quando se tem uma série de instâncias de processos a serem disparadas. Esta carga se refere à freqüência com que os processos são instanciados (instâncias/unidade de tempo). A outra opção define a política de ordenamento/priorização das tarefas.

Finalmente, através da opção “Quantidade de Simulações” é possível definir a quantas instâncias de um processo deverá ser aplicada a simulação.

Após o preenchimento das opções, o usuário pode solicitar o início da simulação. Tal simulação é realizada de acordo com os parâmetros fornecidos, sendo que apenas as informações com relação ao tempo de execução das instâncias processos são apresentadas numa primeira página. Informações mais específicas com relação a cada instância podem ser acessadas de maneira progressiva, de acordo com o interesse do usuário. A Figura 6.6 mostra a primeira página apresentada ao usuário e um nível de detalhe após interação do usuário.

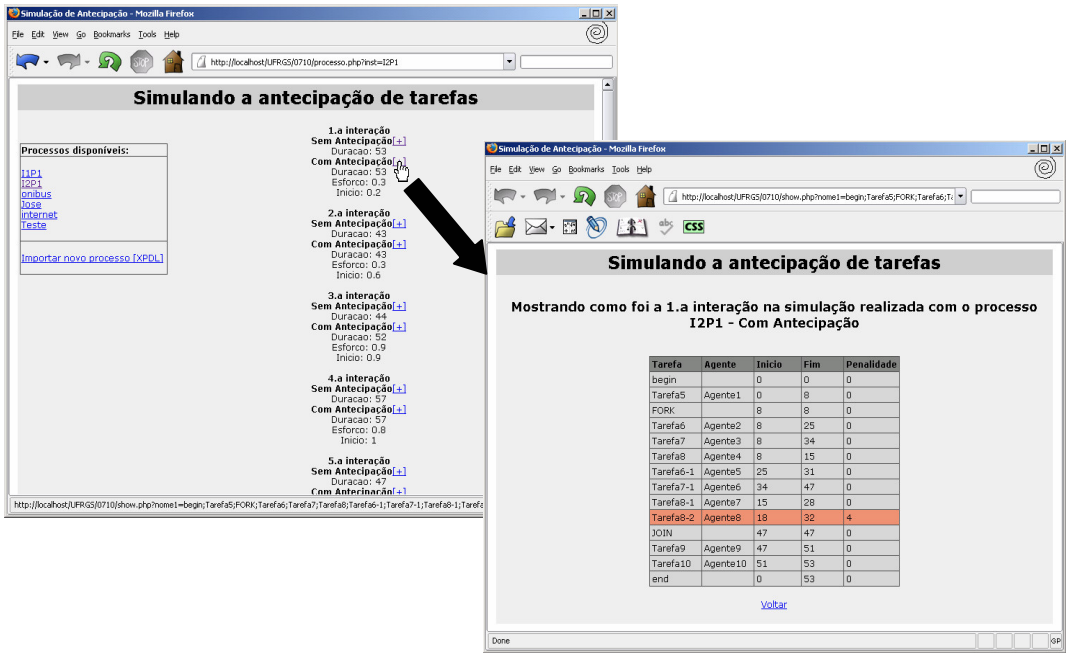

Figura 6.6: Acesso progressivo aos resultados da simulação

A implementação atual apresenta ao usuário as informações apenas de maneira textual. A idéia é a inserção de uma imagem SVG (W3C, 2004b) representando a definição do processo quando aquele processo for selecionado e/ou simulado. Esta imagem apresentaria o fluxo de trabalho e informações relativas à simulação (duração das tarefas, tarefas antecipadas, etc). Para apresentar tais informações, seria necessário escrever um documento XSLT para transformar o processo em XPDL em sua representação gráfica em SVG. O padrão SVG 1.2 (W3C, 2004d) prevê a implementação de fluxos orientados de maneira trivial. Porém, não existem implementações que façam o *parsing* deste padrão, ficando a implementação de tal documento XSLT para um trabalho futuro.

A implementação atual do simulador pode ser acessada e utilizada através do endereço http://cemt.inf.ufrgs.br/wf/simula

6.2 Ferramenta de definição de processos *Amaya Workflow*

A fim de permitir a identificação das tarefas antecipáveis, foram feitas algumas alterações no software de definição de processos *Amaya Workflow* (TELECKEN, 2004). Isto foi feito de maneira simples, através da inclusão de um atributo extra no elemento tarefa.

A definição deste atributo é feita através do menu principal do software, como um novo item no menu *Attributes*. Este novo item se chama *wantecipa* e seus valores podem ser “S” e “N”, correspondendo a Sim (Antecipável) ou Não (Não Antecipável). Ao clicar no menu, uma caixa de texto será apresentada ao usuário que deve preencher sua opção. A Figura 6.7 apresenta o menu com o novo item.

Quando exportado para XPDL, estes dois atributos aparecerão como *ExtendedAttributes* da tarefa à qual estes foram relacionados. O aninhamento deste

atributo dentro de uma tarefa é apresentado no trecho do processo representado em XPDL apresentado na Figura 6.8.

Além deste item foram incluídos os atributos *wdurMin* e *wdurMax*, correspondendo aos valores de duração mínima e máxima de cada tarefa. Estes atributos são utilizados pelo simulador implementado e foram apresentados na seção 6.1.

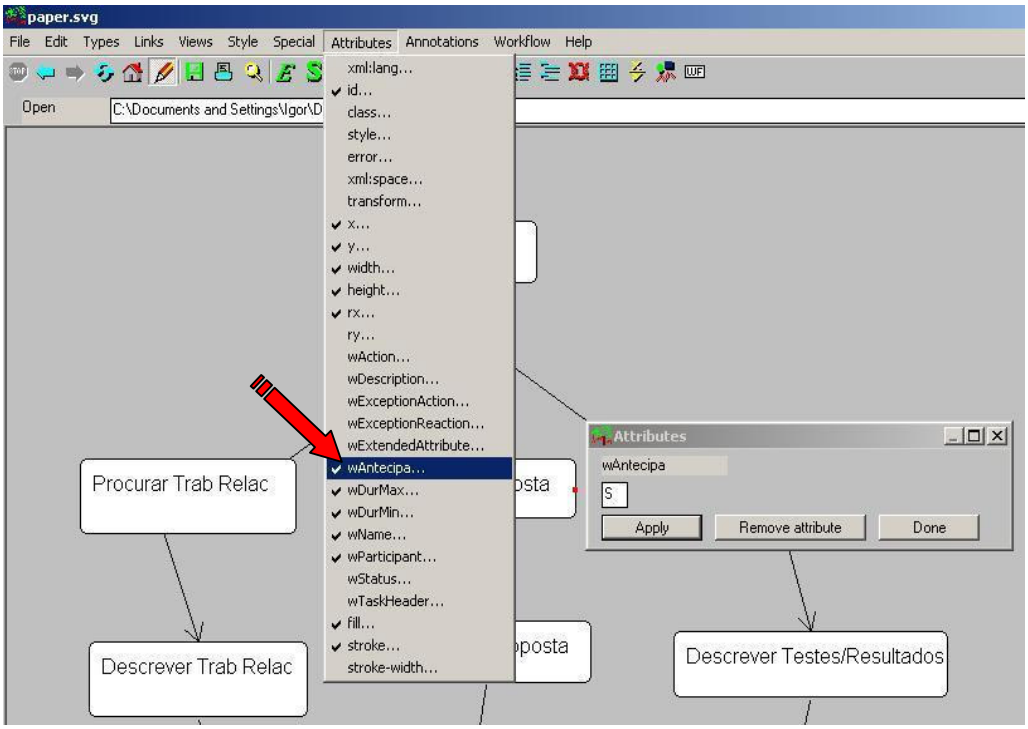

Figura 6.7: Interface da Ferramenta Amaya *Workflow* modificada para modelar tarefas antecipáveis

```
...
 <Activity Name= "Tarefa1" Id= "T1" >
      <Implementation>
          <Tool Id="checkData" Type="APPLICATION">
              <Participant>
                  ...
              </Participant>
              <ActualParameters>
                  ...
              </ActualParameters>
          </Tool>
      </Implementation>
      <ExtendedAttributes>
          <ExtendedAttribute Name="antecipa" value="N"/>
          <ExtendedAttribute Name="durMin" value="8"/>
          <ExtendedAttribute Name="durMax" value="15"/>
      </ExtendedAttributes>
  </Activity>
...
```

Figura 6.8: Trecho de definição de processos explicitando o atributo estendido criado para representar a antecipação de uma tarefa

6.3 Máquina de *Workflow*

Para permitir a utilização da antecipação em casos reais, foi sugerida a implementação dos estados apresentados no capítulo 4 junto à máquina de *workflow* implementada por França (2004), como parte do projeto CEMT (LIMA, 2001). As alterações realizadas no momento apenas dizem respeito à interpretação, armazenamento e utilização do atributo estendido proposto para identificar as tarefas antecipáveis, e aos novos estados que permitem a antecipação de tarefas.

Como o mecanismo de importação da máquina de *workflow* já mapeava os Extended Attributes para o banco de dados relacional, foi simples realizar a distinção entre as tarefas antecipáveis e não antecipáveis. Portanto, foi necessário apenas relacionar este atributo à antecipação na máquina de execução.

As alterações realizadas na máquina de *workflow* estão concentradas em duas funções da WAPI (*Workflow Application Programming Interface*) que já estavam implementadas e na apresentação ao usuário: WMStartProcess e WMChangeActivityInstanceState.

Primeiramente, foi necessário alterar a função WMStartProcess. Esta função é responsável por disparar uma instância de processo criada. Quando executada, esta altera o estado de todas as tarefas para `nãoRodando`, e da primeira tarefa do processo para `rodando`. Justamente neste ponto é que as tarefas são mapeadas aos estados `antecipável` e `nãoAntecipável`, de acordo com o atributo que identifica tal propriedade.

Outra função alterada foi a WMChangeActivityInstanceState. Esta é a função responsável por realizar as alterações dos estados das tarefas. Na realidade, foram alteradas as funções que chamavam esta, a fim de permitir a mudança para os estados previstos para viabilizar a antecipação. Dentre estas alterações está a chamada à função quando do início da execução das tarefas. Sempre que uma tarefa é solicitada para execução, há uma verificação no estado de suas sucessoras e, aquelas antecipáveis, são passadas ao estado `pronta.paraAntecipar`. Foi criada ainda a função que faz o disparo antecipado das tarefas, realizando a transição entre `pronta.paraAntecipar` e `antecipando`.

Evidentemente também foi necessário criar os novos estados relativos à antecipação e alterar a função que faz a checagem das condições necessárias para ativar uma transição.

A apresentação das tarefas ao usuário já traz as características de antecipação. A interface com o usuário permite que seja escolhida aquela tarefa que se deseja executar, dentre as executáveis e antecipáveis. A opção de conclusão da tarefa somente aparece para aquelas tarefas cujas antecessoras foram concluídas, de maneira a seguir o diagrama de estados propostos e manter a dependência *média* ($(fim(t) > fim(t_ant))$).

A Figura 6.9 traz a interface com o usuário com tarefas `antecipáveis`, `pronta.paraAntecipar` e `antecipando`. As setas apontam os novos estados sendo apresentados e a possibilidade de solicitar a antecipação.

Não foram realizados testes com este sistema, pois existem algumas pendências. Para conclusão é necessário, primeiramente, implementar o mecanismo de troca de resultados intermediários. No momento o sistema apenas toma as informações relativas ao fluxo de controle para permitir a antecipação. As

informações relativas ao fluxo de dados não são consideradas ainda. Para que o mecanismo de antecipação possa ser testado em casos reais, é indispensável que se faça o controle de resultados intermediários, e que estes sirvam de restrição para o disparo das tarefas.

O mecanismo de percepção que dá suporte à antecipação de tarefas foi discutido no presente trabalho, porém não foi implementado. Este também é indispensável para que possam ser realizados testes reais utilizando-se antecipação. Pretende-se concluir tais implementações para possibilitar a utilização da antecipação em sua plenitude em processos reais.

CEMT WORKFLOW

Lista de Trabalho de igor

Pacote	Definição do Processo:ID	Código da Instância do Processo	Definição da Atividade: ID	Instância da Atividade	Estado	Ação	
Atividades Rodando							
3	1	1	1	1	executando	Abortar	Completar
3	1	1	2	2	antecipando		
Atividades Prontas							
3	1	1	3	3	pronta.paraAntecipar	Antecipar	
Atividades nãoProntas							
3	1	1	4	4	nãoPronta, antecipável		
3	1	1	5	5	nãoPronta, nãoAntecipavel		
Atividades Fechadas							

Figura 6.9: Interface com o usuário da máquina de *workflow* alterada para suportar a antecipação de tarefas

O sistema resultante das alterações na máquina de *workflow* será utilizado para realização de testes reais com processos que permitam a antecipação de tarefas. Pretende-se verificar os ganhos com relação à qualidade do trabalho e a cooperação, que não foram medidas durante este trabalho. Estes testes serão também utilizados para comparação com as simulações realizadas.

A presente versão da máquina de *workflow* pode ser acessada em http://cemt.inf.ufrgs.br/wf/wf/.

7 CONCLUSÕES

Sistemas Gerenciadores de *Workflow* têm atraído muita atenção nos últimos anos, por serem baseados em um modelo simples que permite definir, executar e monitorar a execução de processos. Mas, na prática estes sistemas não estão muito presentes nas empresas. Talvez uma das razões deste fato é a rigidez e a inflexibilidade dos modelos tradicionais. Isto explica porque foi escolhido o tema de antecipação para a presente dissertação.

Este trabalho então apresenta duas abordagens de antecipação de tarefas que permitem maior flexibilidade nos SGWfs tradicionais. Como vantagem direta do uso abordagens está o aumento no espectro de aplicações que podem ter seus processos gerenciados por SGWfs. Além disso, uma das abordagens apresenta como vantagem o aumento da cooperação entre tarefas, que pode conduzir a produtos finais de melhor qualidade.

As duas abordagens identificadas e apresentadas são classificadas de acordo com o momento em que a antecipação é gerenciada (tempo de definição ou tempo de execução), sendo chamadas *a priori* e *a posteriori*. A abordagem *a priori* apresenta a antecipação gerenciada em tempo de definição dos processos. Ela é estabelecida a fim de tornar a antecipação uma prática possível em qualquer SGWf, apenas propondo uma maneira diferente de modelar as tarefas. É proposta uma nova prática de modelagem em que, apenas alterando a granularidade das tarefas envolvidas na antecipação torna-se possível antecipar as tarefas. Apesar de contar com a vantagem de poder utilizar qualquer SGWf existente, a flexibilidade provida por esta abordagem é pequena. Esta flexibilidade se limita ao ponto em que são realizados os envios de resultados intermediários, sendo o número de troca de resultados pré-definido.

A antecipação gerenciada *a posteriori* permite que o gerenciamento da antecipação seja realizado em tempo de execução, aumentando a flexibilidade inerente aos processos. O custo deste aumento desta flexibilidade é a necessidade de alteração na implementação da máquina de *workflow*. A abordagem *a posteriori* aqui apresentada traz alterações no diagrama de estados das tarefas, baseados na proposta de Grigori (2001), porém com algumas alterações e contribuições.

Foi estabelecida uma maneira de identificar as tarefas antecipáveis durante a definição dos processos. Isto possibilita restringir a execução antecipada apenas a algumas tarefas. Esta identificação é feita de maneira simples, inserindo um novo atributo às tarefas, mapeado para o XPDL como um *Extended Attribute*. Também são definidas novas possíveis dependências de fluxo de controle e de dados e o mapeamento destas dependências para o diagrama de estados. Com relação ao fluxo de dados foi proposta uma estrutura de estados para os elementos de dados. Estes elementos agora podem ser vistos como resultados intermediários instáveis ou estáveis, facilitando a conscientização dos agentes que farão uso destes resultados.

Para avaliar o desempenho da antecipação de tarefas frente à execução tradicional foram conduzidas algumas medições. Estas medições foram realizadas em termos de tempo médio de execução dos processos, através de simulações. Os resultados das simulações permitiram identificar ganhos significativos em favor do uso da antecipação e também padrões de comportamento que podem ser utilizados em testes futuros.

Os resultados demonstram que, para a antecipação ser mais eficiente é necessário que esta seja iniciada antes da metade da execução das tarefas antecessoras. Se o uso dos mecanismos de antecipação não demandarem esforço adicional de maneira alguma, seu uso é vantajoso em quase todos os casos. O processo utilizado como exemplo apresentou ganhos de até 33% (para casos onde as tarefas iniciavam a antecipação com 10% de suas antecessoras concluídas). No caso médio, os ganhos com relação à execução tradicional (baseada na dependência *forte*) aproximaram-se de 18% (caindo de 23 para 19 dias).

Mesmo com esforço adicional, a antecipação mostra-se vantajosa para grande parte dos casos. Sempre que a antecipação iniciou-se antes dos 40% da execução de suas antecessoras, esta apresentou melhor desempenho para valores de esforço que chegavam a duplicar o tempo de execução das tarefas antecipadas.

Outra conclusão a que se chegou é que se antecipando tarefas próximo ao final de suas antecessoras pode trazer perdas, se o esforço necessário para tal antecipação existir. Para antecipação iniciada após 70% do início das antecessoras estas perdas aparecem quando o esforço é superior a 30%.

Além de explicitar os ganhos, as simulações apresentaram padrões de comportamento relacionando início da antecipação, esforço e tempo de execução dos processos. Estes padrões podem ser utilizados para habilitar/desabilitar a antecipação de acordo com o histórico de esforço inserido pela antecipação. Por exemplo, pode-se desabilitar a antecipação de uma tarefa se a antecessora desta já estiver ultrapassado 60% do seu tempo médio de execução (visto o esforço médio – histórico – e o comportamento do processo para antecipação iniciada após 60%).

Em nossas simulações não foram analisados os possíveis ganhos de qualidade do produto final relativo ao aumento na cooperação. Para tal é necessária a realização de testes reais, comparando vários resultados de processos com e sem a utilização de antecipação. Também não foram analisados custos relacionando esforço e tempo de execução. Estes testes estão previstos nos trabalhos futuros.

Todos os testes foram realizados utilizando a ferramenta de definição *Amaya Workflow* modificada para modelar processos prevendo a antecipação e o protótipo de simulador implementado como parte deste trabalho. O protótipo foi implementado para prover meios de simular a antecipação, com a variação de certos eventos (como esforço, início da antecipação e carga de trabalho).

Os resultados conseguidos com as simulações são a maior contribuição da presente dissertação. Além desta contribuição, podem ser enumeradas algumas outras apresentadas no decorrer do trabalho:

- Revisão bibliográfica e classificação dos tipos de flexibilidade oferecidas nas abordagens estudadas (Capítulo 3);
- Identificação e classificação de maneiras diferentes de antecipação de tarefas, oferecendo meios de prover antecipação durante a modelagem ou flexibilizando a máquina de *workflow* (Capítulo 4);

- Análise e dos elementos de percepção necessários para suportar a antecipação de tarefas, visando minimizar o esforço de comunicação e troca de resultados intermediários (Capítulo 5);
- Implementação de um protótipo de simulador com suporte à antecipação que viabilizou a obtenção dos resultados aqui apresentados (Capítulo 8);
- Estabelecimento de variáveis que permitem "simular" o comportamento humano durante a antecipação (Capítulo 7);
- Implementação das funcionalidades de antecipação na Ferramenta de Definição de Processos *Amaya Workflow* (Capítulo 8).
- Modificação da máquina de *workflow* proposta por França (2004) para que esta suporte a antecipação de tarefas (Capítulo 8).

7.1 Trabalhos Futuros

Esta dissertação deixou vários pontos em aberto, podendo estes serem retomados em trabalhos futuros.

O primeiro ponto é analisar a possibilidade de utilizar alguma das abordagens de alteração dinâmica de processos apresentadas no capítulo 3 (JOERIS; HERZOG, 1998; 1999; REICHERT; RINDERLE; DADAM, 2003) para prover antecipação de tarefas. Estas abordagens poderiam ser modificadas de maneira a criar tarefas automaticamente durante a execução dos processos, quando fossem fornecidos resultados intermediários. Isto permitiria que a antecipação fosse gerenciada *a posteriori*, porém, sem alteração no diagrama de transição de estados.

Outro ponto está relacionado à troca de dados entre as tarefas. É necessário que se defina um protocolo de cooperação para oferecer meios seguros para a troca de resultados intermediários. Este protocolo deve incluir estudos e proposta de um mecanismo de controle de versões e mecanismos de resolução de conflitos.

A conclusão da implementação da máquina de *workflow* que permita a antecipação de tarefas é outro trabalho futuro. Esta implementação permitirá validar as simulações realizadas, bem como extrair informações mais ricas das simulações. Será possível verificar o comportamento dos usuários, e o esforço adicional que a antecipação impõe à execução.

A conclusão do protótipo compreende o item anterior e também a implementação do mecanismo de percepção que suporte a antecipação. No trabalho são apenas identificados e analisados alguns elementos que o mecanismo deve contemplar. Estes devem ser implementados e sua utilidade verificada em testes reais.

7.2 Publicações

Além das contribuições apresentadas acima, também fazem parte dos resultados desta dissertação as seguintes publicações:

STEINMACHER, I.; LIMA, J.V. Melhorando a Cooperação Através da Antecipação de Tarefas em *Workflow*. In: WEBMEDIA AND LA-WEB JOINT

CONFERENCE; WEBMEDIA PHD AND MSC PROPOSALS WORKSHOP, 2004, Ribeirão Preto. **Proceedings…** Ribeirão Preto: COC, 2004. p. 209-212.

STEINMACHER, I.; LIMA, J.V.; FREITAS, A.V.P. Uma Maneira de Antecipar Tarefas em um Sistema Gerenciador de *Workflow* In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN (CACIC), 2004, Buenos Aires, Argentina. **Proceedings...** [S.l.: s.n.], 2004.

TELECKEN, T.; STEINMACHER, I.; LIMA, J.V. Amaya *Workflow*: Uma Ferramenta de definição de processos. In: WEBMEDIA AND LA-WEB JOINT CONFERENCE; WEBMEDIA DEMOS AND TOOLS WORKSHOP, 2004, Ribeirão Preto. **Anais…** Ribeirão Preto: COC, 2004. p. 295-296.

STEINMACHER, I.; VIDAL, N.R.; TELECKEN, T.; LIMA, J.V. Proposta de sistema de agendamento de tarefas para *workflow* baseado em prioridades. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 4., 2004, Itajaí. **Anais...** Itajaí: UNIVALI, 2004. p. 143-148.

A Submeter:

STEINMACHER, I.; LIMA, J.V. Using task anticipation within a *Workflow* Management System: a Quantitative Analysis.

REFERÊNCIAS

AALST, W.; HEE, K. ***Workflow* Management:** Models, Methods and Systems. Massachussets: The MIT Press, 2002.

AIELLO, R. ***Workflow* Performance Evaluation**. 2004. Tese (Doutorado em Informática) – Università di Salerno, Salerno.

ALLEN, J.F. Maintaining knowledge about temporal intervals. **Communications of the ACM**, New York, v. 32, n. 11, p. 832-843, Nov. 1983.

ARAUJO, R.M. **Ampliando a Cultura de Processos de Software – Um enfoque Baseado em Groupware e *Workflow***. 2000. Tese (Doutorado em Computação) – COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

CANALS, G.; GODART, C.; MOLLI, P.; MUNIER, M. A criterion to enforce correctness of indirectly cooperation applications. **Information Sciences:** an International Journal, New York, v. 110, n. 3-4, p. 279-302, Oct. 1998.

CASATI, F. et al. Conceptual modeling of *workflow*. In: INTERNATIONAL CONFERENCE ON CONCEPTUAL MODELING, ER, 1995, Gold Cost, Australia. **Conceptual Modeling**: proceedings. Berlin: Springer Verlag, 1995.

CASATI, F.; GREFEN, P.; PERNICI, B.; POZZI, G.; SÁNCHEZ, G. **WIDE *Workflow* Model and Architecture**. Netherlands: Centre for Telematics and Information Technology (CTIT), University of Twente, 1996. (Technical Report, 96-19).

DOURISH, P.; HOLMES, J.; MACLEAN, A.; MARQVARDSEN, P.; ZBYSLAW, A. Freeflow: Mediating Between Representation and Action in *Workflow* Systems. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 1996, Boston, EUA. **Proceedings…** New York: ACM Press, 1996. p. 190-198.

ELLIS, C.; KEDDARA, K.; ROZENBERG, G. Dynamic Change Within *Workflow* Systems. In: INTERNATIONAL CONFERENCE ON ORGANIZATIONAL COMPUTER SYSTEMS, 1995, EUA. **Proceedings...** New York: ACM Press, 1995. p. 10-21.

FRANÇA, M. B. **Proposta e implementação de uma máquina de *workflow* para o projeto CEMT**. 2004. 97f. Dissertação (Mestrado em Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

FUKS, H.; GEROSA, M.A.; PIMENTEL, M.G. Projeto de Comunicação em Groupware: Desenvolvimento, Interface e Utilização. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, JAI, 22., 2003, Campinas. **Anais...** Campinas: SBC, 2003. p. 295-338.

FUSSELL, S.R.; KRAUT, R.; LEARCH, F.; SCHERLIS, W.; MCNALLY, M.; CADIZ, J.J. Coordination, overload and team performance: effects of team communication strategies. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, CSCW, 1998, Seattle, EUA. **Proceedings**… New York: ACM Press, 1998. p.275-284.

GODART, C.; PERRIN, O.; SKAF, H. Coo: a *Workflow* Operator to Improve the Cooperation Modeling in Virtual Processes. In: INTERNATIONAL WORKSHOP ON RESEARCH ISSUES ON DATA ENGINEERING: INFORMATION TECHNOLOGY FOR VIRTUAL ENTERPRISES, 9., 1999, Sidney, Austrália. **Proceedings…** Washington: IEEE Computer Society, 1999. p. 126-131.

GODART, C.; MOLLI, P.; OSTER, G.; PERRIN, O.; SKAF-MOLLI, H. The ToxicFarm Integrated Cooperation Framework for Virtual Teams. **Distributed and Parallel Databases:** Special Issue on Teamware, Hingham, EUA, v.15, n.2, p. 67-88, Jan. 2004.

GRIGORI, D. **Eléments de flexibilité des systèmes de *workflow* pour la définition et l'exécution de procédés coopératifs**. 2001. 135f. Tese (Doutorado em Informática) – Université Henri Poincaré, Nancy.

GRIGORI, D.; CHAROY, F.; GODART, C. Flexible Data Management and Execution to Support Cooperative *Workflow*: the COO approach. In: INTERNATIONAL SYMPOSIUM ON COOPERATIVE DATABASE SYSTEMS FOR ADVANCED APPLICATIONS, 3., 2001, Beijing, China. **Proceedings…**Washington: IEEE Computer Society, 2001. p. 124-131.

GRIGORI, D.; CHAROY, F.; GODART, C. Coo-Flow: A Process Technology to Support Cooperative Processes. **International Journal of Software Engineering and Knowledge Engineering**, [S.l.], v. 14, n.1, p. 1-19, Jan. 2004.

HAGEN, C.; ALONSO, G. Beyond the black box: Event-based inter-process communication in process support systems. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS,19. , Austin, EUA, 1999. **Proceedings…** Washington: IEEE Computer Society, 1999. p. 450–457.

HEINL, P.; HORN, S.; JABLONSKI, S.; NEEB, J.; STEIN, K.; TESCHKE, M. A Comprehensive Approach to Flexibility in *Workflow* Management Systems. In: INTERNATIONAL JOINT CONFERENCE ON WORK ACTIVITIES COORDINATION AND COLLABORATION, 1999, San Francisco, EUA. **Proceedings…** New York: ACM Press, 1999. p. 79-88.

JOERIS, G.; HERZOG, O. **Towards Object-Oriented Modeling and Enacting of Processes**. Bremen: Technologie-Zentrum Informatik, Universität Bremen, 1998.

JOERIS, G. Defining Flexible *Workflow* Execution Behaviors. In: ENTERPRISE-WIDE AND CROSSENTERPRISE *WORKFLOW* MANAGEMENT: CONCEPTS,

SYSTEMS, APPLICATIONS, 1999, Ulm, Alemanha. **Proceedings…** [S.l.: s.n.], 1999.

JOERIS, G.; HERZOG, O. Towards Flexible and High-Level Modeling and Enacting of Processes. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING, CAiSE, 11., 1999, Heidelberg, Alemanha. **Proceedings…** Londres: Springer Verlag, 1999. p. 88-102.

KRAMLER, G.; RETSCHITZEGGER, W. Towards Intelligent Support of *Workflow*. In: AMERICAS CONFERENCE ON INFORMATION SYSTEMS, AMCIS, 2000, Long Beach, EUA. **Proceedings…** [S.l.: s.n.], 2000. p.581-585.

LAGUNA, M.; MARKLUND, J. **Business Process Modeling, Simulation, and Design.** Upper Saddle River: Prentice Hall, 2004.

LIMA, J. V.; QUINT, V.; LAYAIDA, N.; EDELWEISS, N.; ZEVE, C. M. D.; PINHEIRO, M. K.; TELECKEN, T. L. The Conception of Cooperative Environment for Editing Multimedia Documents with *Workflow* Technology (CEMT). In: PROTEM-CC, 4., 2001, Rio de Janeiro. **Projects Evaluation Workshop**: international cooperation: proceedings. Brasília: CNPQ, 2001. p. 542-560.

MANGAN, P.; SADIQ, S. On Building *Workflow* Models for Flexible Processes. In: AUSTRALASIAN CONFERENCE ON DATABASE TECHNOLOGIES, 13., 2002, Melbourne, Austrália. **Proceedings...** Darlinghurst: Australian Computer Society, 2002. p. 103-109.

MYSQL. **Reference Manual for Version 4.0.14**. Disponível em: <http://www.mysql.com/downloads/download.php?file=Downloads%2FManual%2Fmanual.tar.gz&pick=mirror>. Acesso em: 2004.

NICKERSON, J.V. Event-based *Workflow* and the Management Interface. In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 36., 2003, Big Island, Havaí. **Proceedings…** Washington: IEEE Computer Society, 2003.

KIRSCH-PINHEIRO, M.; LIMA, J.V.; BORGES, M.R.S. Awareness em Sistemas de Groupware. In: JORNADAS IBEROAMERICANAS DE INGENIERÍA DE REQUISITOS Y AMBIENTES DE SOFTWARE, 4., 2001, Santo Domingo. **Memorias.** Costa Rica: CIT, 2001. p.323-335.

RAPOSO, A.B.; MAGALHÃES, L.P, RICARTE, I.L.; FUKS, H. Coordination of Collaborative Activities: A Framework for Definition of Tasks Interdependencies. In: INTERNATIONAL WORKSHOP ON GROUPWARE, CRIWG, 7., 2001, Darmstadt, Alemanha. **Proceedings…** [S.l.]: IEEE Computer Society Press, 2001. p. 170-179.

REICHERT, M.; DADAM, P. $\text{Adept}_{\text{flex}}$ – Supporting Dynamic Changes of *Workflow* Without Loosing Control. **Journal of Intelligent Information System:** Special Issue on *Workflow* Management Systems, [S.l.], v.10, n.2, p. 93-129, Mar./Apr. 1998.

REICHERT, M.; RINDERLE, S.; DADAM, P. ADEPT *Workflow* Management System: Flexible Support for Enterprise-Wide Business Processes. In: BUSINESS PROCESS MANAGEMENT INTERNATIONAL CONFERENCE, BPM, 2003, Eindhoven, Holanda. **Proceedings…** Berlin: Springer-Verlag, 2003a. p. 370-379.

REICHERT, M.; RINDERLE, S.; DADAM, P. On the Common Support of *Workflow* Type and Instance Changes Under Correctness Constraints. In: INTERNATIONAL CONFERENCE ON COOPERATIVE INFORMATION SYSTEMS, CoopIS, 2003, Catania, Itália. **Proceedings…** New York: Springer Verlag, 2003b. p. 407-425.

REICHERT, M.; DADAM, P.; BAUER, T. Dealing with Forward and Backward Jumps in *Workflow* Management Systems. **Journal Software and Systems Modeling (SoSyM)**, [S.l.], v. 2, n. 1, p. 37-58, 2003.

RIBÓ, J.M.; FRANCH, X. A Precedence-based Approach for Proactive Control in Software Process Modeling. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING, SEKE, 14., 2002, Ischia, Itália. **Proceedings…** New York: ACM Press, 2002. p. 457-464.

STEINMACHER, I.; LIMA, J.V.; FREITAS, A.V.P. Uma Maneira de Antecipar Tarefas em um Sistema Gerenciador de *Workflow* In: CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN, CACIC, 2004, Buenos Aires, Argentina. **Proceedings...** [S.l.: s.n.], 2004a.

STEINMACHER, I.; LIMA, J.V. Melhorando a Cooperação Através da Antecipação de Tarefas em *Workflow*. In: WEBMEDIA AND LA-WEB JOINT CONFERENCE; WEBMEDIA PHD AND MSC PROPOSALS WORKSHOP, 2004, Ribeirão Preto. **Proceedings…** Ribeirão Preto: COC, 2004b. p. 209-212.

STEINMACHER, I.; VIDAL, N.R.; TELECKEN, T.; LIMA, J.V. Proposta de sistema de agendamento de tarefas para *workflow* baseado em prioridades. In: CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 4., 2004, Itajaí. **Anais...** Itajaí: UNIVALI, 2004c. p. 143-148.

TELECKEN, T. **Um Estudo sobre Modelos Conceituais para Ferramentas de Definição de Processos de *Workflow*.** 2004. 118 f. Dissertação (Mestrado em Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2004.

TRAMONTINA, G. B. **Análise de Problemas de Escalonamento de Processos em *Workflow***. 2004. 64 f. Dissertação (Mestrado em Computação) – Instituto de Computação, Universidade Estadual de Campinas, Campinas.

VAN KOOTEN, J.; MEAGHER, P. **PHP SpecialMath Library**. Disponível em: <http://www.phpmath.com/files?op=viewVersion&fldr=51&fid=129&vid=46>. Acesso em: 2004.

W3C. **Amaya Home Page**. Disponível em: <http://www.w3.org/Amaya/>. Acesso em: 2004a.

W3C. **Scalable Vector Graphics (SVG) 1.0 Specification**. W3C recommendation. 2001. Disponível em: <http://www.w3.org/TR/SVG/>. Acesso em: 2004b.

W3C. **eXtensible Stylesheet Language (XSL) 1.0 Specification**. W3C recommendation. 2001. Disponível em: <http://www.w3.org/TR/XSL/>. Acesso em: 2004c.

W3C. **Scalable Vector Graphics (SVG) 1.2. W3C Working Draft**. 2004. Disponível em: <http://www.w3.org/TR/SVG/>. Acesso em: 2004d.

WFMC. ***Workflow* Management Coalition Home Page**. Disponível em: <http://www.wfmc.org>. Acesso em: 2004.

WFMC. ***Workflow* Management Coalition Interface 1:** process definition interchange process model. [S.l.], 1999. Technical Report WFMC-TC-1016).

WFMC. ***Workflow* Management Coalition Terminology & Glossary**.[S.l.], 1999a (WFMC-TC-1011, Document Status- Issue 3.0).

APÊNDICE A AUMENTANDO A COOPERAÇÃO ATRAVÉS DA ANTECIPAÇÃO DE TAREFAS

Segundo Fuks, Raposo e Pimentel (2003), cooperação é a operação conjunta dos membros do grupo no espaço compartilhado. Os indivíduos cooperam produzindo, manipulando e organizando informações, bem como construindo e refinando artefatos digitais, como documentos, planilhas, gráficos, etc. O aumento da cooperação previsto por esta abordagem diz respeito justamente à possibilidade da troca de resultados intermediários entre as tarefas. Através da troca de resultados intermediários será possível que os agentes compartilhem um mesmo elemento de dado, podendo interagir através deste.

A.1 Mecanismos de trocas de resultados entre tarefas

Existem duas possíveis arquiteturas que podem ser adotadas para a troca de resultados entre os agentes: Arquitetura centralizada e replicada. Estas duas arquiteturas são utilizadas para identificar duas diferentes maneiras de cooperação: Cooperação Indireta e Cooperação Direta. Na arquitetura centralizada, ocorre a cooperação indireta, pois existe um servidor central responsável pelo armazenamento e gerenciamento dos elementos de dados. Quando um agente deseja cooperar ou fornecer algum elemento de dado para haver cooperação, isto deve ser feito através do servidor.

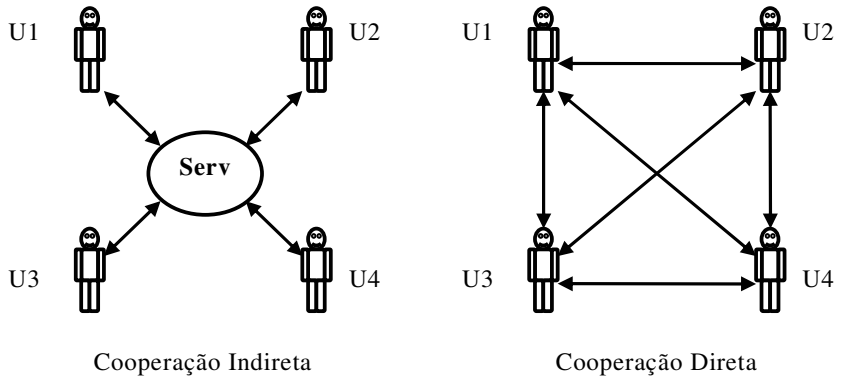

Figura A.1: Possíveis arquiteturas para distribuição dos dados

Na arquitetura replicada, ocorre a cooperação direta, pois cada agente armazena e gerencia os elementos de dados relacionados às suas tarefas. Cada agente seria um cliente e um servidor, ficando responsável por armazenar e gerenciar os elementos de dados localmente. Qualquer alteração em um elemento de dado deve

ser informada e enviada aos outros agentes que possuem uma cópia deste elemento. A Figura A.1 apresenta as duas arquiteturas possíveis.

Segundo Canals *et al.* (1998) as principais características de aplicações que são cooperativas indiretas são:

1. *as tarefas são cooperativas e não competitivas*, isto é, as tarefas interagem a fim de tirar vantagem disto e alcançar um objetivo comum;
2. *as tarefas são assincronamente cooperativas*, isto é, as interações entre tarefas não são previamente programadas;
3. *as tarefas são interativas*, onde o controle está predominantemente sob responsabilidade de humanos;
4. as tarefas têm seu *desenvolvimento incerto*, a execução das tarefas não é totalmente definida;
5. *as tarefas são de longa duração*, podendo durar horas, dias, meses ou anos;
6. *as decisões são reversíveis*.

Mais um argumento a favor da cooperação indireta é a arquitetura de referência da WfMC, que está sendo seguida no presente trabalho. Nesta arquitetura é previsto o uso de um repositório central de dados de aplicação, acessado pelas aplicações invocadas tanto pela máquina de *workflow* quanto pelos usuários do sistema.

Tendo em vista a utilização deste tipo de cooperação, é apresentada uma arquitetura contendo dois níveis de repositórios de dados. Num primeiro nível existe um repositório central, responsável por armazenar e gerenciar todos os dados produzidos pelos agentes e tarefas. E num segundo nível existem os repositórios privados de cada um dos agentes. Nestes repositórios ficam todos os elementos de dados solicitados e utilizados pelo agente a fim de possibilitar a execução de suas tarefas.

Ambos repositórios (central e privado) ficam em um servidor acessível de qualquer local a qualquer momento pelos agentes. Isto colabora com a mobilidade provida pelo ambiente, já que o usuário pode acessar seus dados de qualquer lugar, a qualquer momento, possibilitando trabalho remoto. Segundo Fuks, Gerosa e Pimentel (2003) existe a necessidade de independência do ambiente com relação à máquina onde ele está sendo executado. Quando este requisito for atendido, os usuários terão o ambiente da mesma forma que o deixaram, mesmo que ele esteja sendo utilizado em uma máquina diferente.

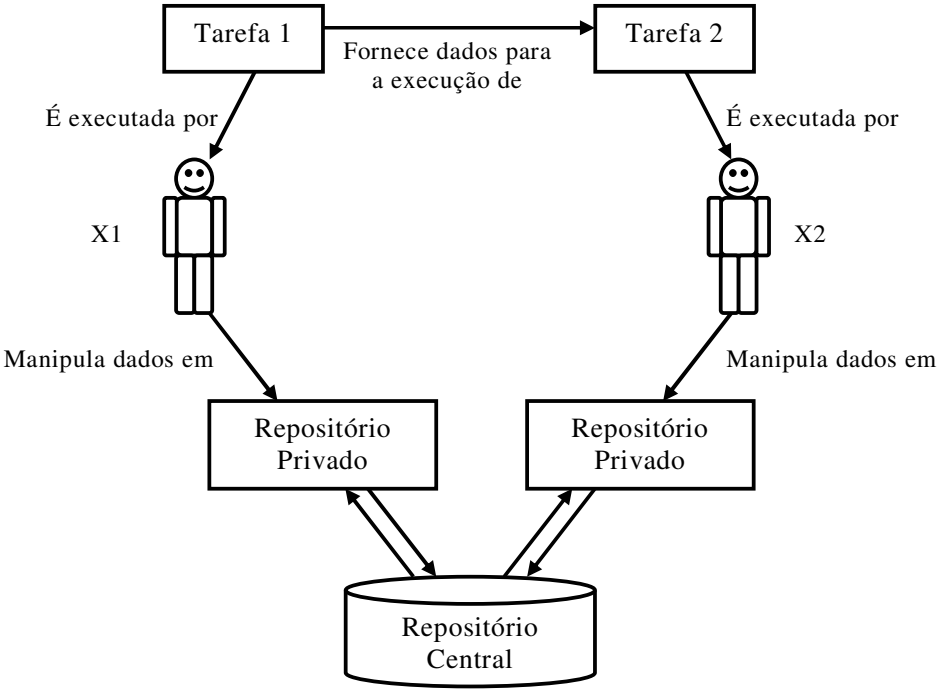

Figura A.2: Arquitetura com dois níveis de repositório para possibilitar a troca de resultados entre tarefas

A Figura A.2 apresenta a arquitetura com dois níveis de repositórios. Um agente (X2), está executando uma determinada tarefa (Tarefa 2), que depende de dados de uma outra tarefa (Tarefa 1), que está sendo executada por outro agente (X1). Cada um dos agentes possui um repositório privado onde armazena os dados referentes às tarefas sendo executadas por ele. Quando X1 fornece algum dado relevante à Tarefa 2, X2 pode solicitar a leitura de tal dado, que é colocado em seu repositório privado. No caso da figura, apenas X2 lerá os dados fornecidos por X1, não havendo necessidade de sincronização dos resultados. Quando for necessário (quando houver uma relação de *feedback*) o servidor é responsável por realizar a sincronização dos resultados, informando aos agentes envolvidos sobre possíveis inconsistências.

A.2 Estratégia de atualização dos elementos de dados – *push x pull*

Outro ponto importante com relação aos elementos de dados é a decisão pela estratégia de atualização dos objetos que estão sendo manipulados pelos agentes. Por exemplo, uma tarefa que está sendo executada de maneira antecipada está utilizando um resultado intermediário de uma antecessora. Quando esta antecessora libera um outro resultado deste mesmo elemento de dado, é necessário que este seja atualizado no repositório privado do agente que está utilizando o resultado intermediário. Para tal, duas possíveis estratégias podem ser utilizadas: *pull* e *push*.

Através da estratégia *push* os dados são "empurrados" para os repositórios daqueles agentes que estão fazendo uso do resultado intermediário, sendo estes atualizados automaticamente, sempre que houver a liberação de um novo resultado.

Já com a estratégia *pull* os agentes são apenas informados da existência de novos resultados, ficando a cargo destes fazer o "*download*" das novas versões e atualizá-las em seu espaço particular. A atualização por parte do agente não é obrigatória para todas as versões, podendo este apenas buscar e atualizar as versões que lhe interessam, no momento que escolher. A única restrição deve-se ao resultado final, que deve (obrigatoriamente) ser buscado pelo agente, para que seja possível finalizar a execução de sua tarefa e a produção de seu resultado final.

No presente trabalho é considerada a utilização da estratégia *pull* a fim de deixar que os agentes decidam pelo melhor momento de recuperar as versões do resultado. Isto é feito de maneira a tornar o trabalho cooperativo mais flexível, não obrigando os agentes utilizarem todos os resultados. Sempre que houver um novo resultado de um dos elementos de dados utilizados pelo agente, este será avisado. Quando o usuário decidir pela "leitura" de um novo resultado, o resultado será enviado e atualizado.