Mecanismos de Coordenação para Ambientes Colaborativos

Alberto B. Raposo, Léo P. Magalhães, Ivan L. M. Ricarte
Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
{alberto, leopini, ricarte}@dca.fee.unicamp.br

Resumo. A coordenação das interdependências entre atividades colaborativas é uma tarefa complexa, de difícil modelagem computacional. Neste trabalho é apresentado um conjunto de mecanismos de coordenação para a especificação e controle da interação entre tarefas colaborativas. Os mecanismos são modelados por redes de Petri e permitem avaliar o comportamento de um ambiente de suporte ao trabalho colaborativo antes de sua implementação.

Abstract. The coordination of interdependencies among collaborative activities is a complex task, not easily computer modeled. In this paper, a set of coordination mechanisms for the specification and control of interaction among collaborative tasks is presented. The mechanisms are modeled using Petri nets and enable to evaluate the behavior of a computer supported collaborative work system before its implementation.

1. Introdução

Trabalho colaborativo foi definido por Marx como "múltiplos indivíduos trabalhando juntos de maneira planejada no mesmo processo de produção ou em processos de produção diferentes, mas conectados" [Bannon 91]. No âmago desta definição está a noção de planejamento, responsável por garantir que o trabalho coletivo seja resultante do conjunto de tarefas individuais.

No entanto, algumas atividades envolvendo múltiplos indivíduos não exigem um planejamento formal. Atividades relacionadas às relações sociais são bem controladas pelo chamado protocolo social, caracterizado pela ausência de qualquer mecanismo de coordenação entre as atividades, confiando nas habilidades dos participantes de mediar as interações. Exemplos de atividades deste tipo em computador são os *chats* e as áudio- ou vídeo-conferências.

Por outro lado, atividades mais diretamente voltadas para o *trabalho* colaborativo (e não para as relações sociais) exigem sofisticados mecanismos de controle para evitar que os participantes se envolvam em tarefas conflitantes e/ou repetitivas. É este tipo de atividade que tem sido o principal alvo dos esforços de pesquisa em CSCW (*Computer Supported Cooperative Work*), uma área que estuda "a natureza e as características do trabalho cooperativo com o objetivo de projetar tecnologias computacionais adequadas" [Bannon 91].

A noção de planejamento presente na definição de Marx é materializada em CSCW pelo que foi chamado de trabalho de articulação, que é o esforço extra necessário para que a colaboração seja obtida a partir da soma dos trabalhos individuais. Fazem parte do trabalho de articulação a identificação dos objetivos, o mapeamento destes objetivos em tarefas, a

seleção dos participantes, a distribuição das tarefas entre eles e a coordenação da realização das atividades.

Particularmente importante entre as atividades do trabalho de articulação é a coordenação, definida como "o ato de gerenciar interdependências entre as atividades realizadas para se atingir um objetivo" [Malone 90]. A coordenação representa o aspecto dinâmico do trabalho de articulação, pois precisa ser renegociada de maneira quase contínua ao longo de todo o tempo que durar a colaboração.

A coordenação de atividades em ambientes colaborativos auxiliados por computador é o principal foco deste artigo, que apresenta um conjunto de mecanismos de coordenação para uma série de interdependências que freqüentemente ocorrem entre tarefas colaborativas. A idéia é separar as atividades (tarefas) das dependências (controladas pelos mecanismos de coordenação), permitindo o uso de diferentes políticas de coordenação em um mesmo ambiente colaborativo, sendo necessário apenas trocar os mecanismos de coordenação. Além disso, torna-se possível o reuso dos mecanismos de coordenação em outros ambientes colaborativos. Para modelar os mecanismos de coordenação propostos, é usada uma abordagem baseada em redes de Petri (PNs – *Petri Nets*).

A próxima seção mostra uma visão geral de trabalhos relacionados à coordenação de atividades colaborativas. A Seção 3 faz uma breve introdução às PNs, e a Seção 4 apresenta os mecanismos desenvolvidos. Um exemplo de uso dos mecanismos de coordenação em uma situação de CSCW é mostrado na Seção 5. A Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Apesar de sua reconhecida importância, mecanismos de coordenação não foram incluídos nos primeiros sistemas colaborativos. As tentativas de incluir mecanismos de coordenação neste tipo de sistema datam do final dos anos 80. Um dos mais significativos representantes da primeira geração de sistemas colaborativos com algum mecanismo de coordenação foi o *Coordinator* [Winograd 86], desenvolvido com base em teorias lingüísticas, com o objetivo de ajudar a estruturação de conversas por e-mail. O *Coordinator* segue padrões de conversação pré-estabelecidos para direcionar a realização das tarefas.

Desde aquela época há exemplos de uso de PNs para a coordenação de atividades em ambientes colaborativos [Holt 85], [De Cindio 88].

Na década de 90 começaram a surgir sistemas que tentam tornar os mecanismos de coordenação mais flexíveis e acessíveis aos usuários. Um destes sistemas é o Oval [Malone 95], que usa o conceito de EUP (*End User Programming*) para permitir que o usuário (não necessariamente um programador experiente) modifique o sistema sem sair do domínio da aplicação e ir para o domínio de uma linguagem de programação.

Inspirados pelas linguagens de coordenação [Gelernter 92], que propuseram a separação computação/coordenação para aplicações *multi-threaded*, surgiram sistemas colaborativos que separam a implementação dos componentes de coordenação das demais partes dos mesmos. Isso permite maior flexibilidade no uso de políticas de coordenação. O COCA (*Collaborative Objects Coordination Architecture*) [Li 98] e o Trellis [Furuta 94] são exemplos de sistemas deste tipo. O Trellis, em particular, usa uma variação de PNs em um servidor para especificar protocolos de interação para um grupo de clientes.

PNs também têm sido usadas para modelar sistemas de *workflow*, definidos como um "tipo particular de *groupware* para auxiliar grupos de pessoas na execução de procedimentos de trabalho; ele possui o conhecimento de como o trabalho normalmente flui em uma organização" [Ellis 93]. O COSM (*Common Open Service Market*) [Merz 97], por exemplo,

usa agentes móveis para gerenciar *workflows* interorganizacionais, onde cada organização tem seu esquema interno de controle para a realização das tarefas (*workflow*), mas precisa colaborar com outras organizações [WfMC 96]. O comportamento destes agentes é controlado por uma máquina virtual modelada em PN.

PNs também foram usadas para o planejamento de animações interativas, de forma a prever seu comportamento antes de gerar as imagens [Magalhães 98].

O trabalho aqui apresentado é mais genérico que os descritos acima, pois define um conjunto de interdependências entre tarefas e mecanismos de coordenação associados (modelados por PNs) que podem ser usados em *workflows*, trabalho cooperativo, interação multiusuário, ambientes virtuais, etc. O objetivo não é implementar um sistema fechado, mas prover uma série de mecanismos que permitam ao projetista de um ambiente colaborativo prever e testar seu comportamento antes de implementá-lo, detectando as possíveis falhas da muitas vezes complexa cadeia de interação entre as atividades.

3. Redes de Petri

Rede de Petri é uma ferramenta de modelagem aplicável a uma série de sistemas, especialmente aqueles com eventos concorrentes. Formalmente, uma PN é definida como uma quíntupla (P, T, F, w, M_0) onde: $P = \{P_1, ..., P_m\}$ é um conjunto finito de lugares (*places);* $T = \{t_1, ..., t_n\}$ é um conjunto finito de transições; $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos; $w: F \rightarrow \{1, 2, ...\}$ é uma função que dá peso aos arcos; $M_0: P \rightarrow \{1, 2, ...\}$ é a marcação inicial da rede (número de *tokens* em cada lugar); com $(P \cap T) = \varnothing$ *e* $(P \cup T) \neq \varnothing$.

No modelo de PN, os estados estão associados aos lugares e suas marcações, e os eventos às transições. Uma transição t está habilitada se cada um de seus lugares de entrada P_i possuir pelo menos $w(P_i, t)$ *tokens*, onde $w(P_i, t)$ é o peso do arco ligando P_i a t. Estando habilitada, uma transição pode ser disparada quando o evento associado a ela ocorrer. O disparo de t remove $w(P_i, t)$ *tokens* de cada um de seus lugares de entrada P_i e adiciona $w(t, P_o)$ *tokens* a cada lugar de saída P_o.

A notação gráfica de PNs é também muito utilizada (Figura 1). Nesta notação, os lugares são representados por círculos, as transições por barras ou retângulos, os *tokens* por pontos, e os arcos por setas com os pesos escritos em cima (por definição, um arco não marcado tem peso 1).

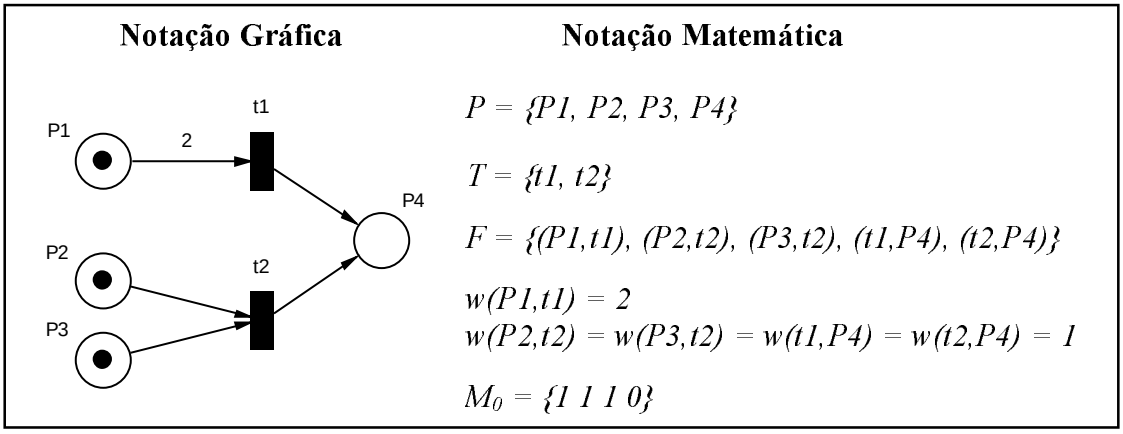

Figura 1: Notação gráfica e notação matemática de PNs.

Além do modelo básico, existem várias extensões de PN [Murata 89]. Duas extensões são utilizadas neste trabalho: arco inibidor e redes com tempo. O arco inibidor liga um lugar P a

uma transição *t* e a habilita apenas se *P* estiver vazio. Na notação gráfica, arcos inibidores são representados com um círculo na extremidade. O modelo básico de PN não considera a noção de tempo. Uma maneira de incluir esta noção em uma PN é definir um tempo para o disparo das transições. Neste caso, os *tokens* são retirados dos lugares de entrada e algum tempo depois (tempo de disparo) são entregues aos lugares de saída. Este tipo de disparo não-instantâneo é também chamado de disparo com reserva de *tokens*.

As PNs também oferecem importantes ferramentas para análise do sistema modelado. Há três tipos possíveis de análise: verificação, validação e desempenho [van der Aalst 98]. A análise de verificação é realizada para garantir que a rede esteja corretamente definida e corresponda com exatidão ao sistema modelado. Neste tipo de análise é verificado se a rede apresenta *deadlocks*, se atinge algum estado não permitido, se há transições mortas, etc. A análise de validação testa se a rede funciona como esperado. Os testes são feitos por meio de simulação iterativa de situações fictícias para verificar se a rede as trata corretamente. A análise de desempenho avalia a capacidade do sistema atingir certos requisitos, tais como tempo médio de espera, número médio de casos pendentes, uso de recursos, *throughput times*, etc.

Em resumo, pela capacidade de modelagem, forte suporte teórico para a análise e grande número de técnicas de simulação, PNs são ferramentas adequadas para o planejamento e coordenação de atividades colaborativas. Por esta razão elas são usadas como ferramenta de modelagem dos mecanismos de coordenação apresentados na próxima seção.

4. Mecanismos de Coordenação

Este trabalho trata da inserção de mecanismos de coordenação para gerenciar interdependências entre tarefas e garantir que estas dependências não sejam violadas. A idéia é fazer com que o projetista do ambiente se preocupe apenas com a definição da rede que modela as tarefas e com a definição das interdependências entre estas tarefas, não com os mecanismos para gerenciar estas dependências.

No esquema proposto, o ambiente colaborativo é modelado em dois níveis hierárquicos distintos: nível de *workflow* e nível de coordenação. No nível de *workflow* são definidos o seqüenciamento das tarefas e as dependências entre elas. No nível de coordenação, as tarefas interdependentes são expandidas e os mecanismos de coordenação são inseridos entre elas.

Durante a passagem do nível de *workflow* para o nível de coordenação, cada tarefa (representada por uma transição no nível de *workflow*) que possuir uma interdependência com outra tarefa é expandida em um sistema como o da Figura 2 [van der Aalst 94]. Os cinco lugares associados às transições (*request_resource*, *assigned_resource*, *execute_task*, *finish_task* e *release_resource*) representam a interação com o gerenciador de recursos e com o agente que realiza a tarefa. O *request_resource* indica ao gerenciador que a tarefa deseja um determinado recurso. Após a alocação deste recurso, o gerenciador coloca um *token* em *assigned_resource*, para dar prosseguimento à tarefa. Os lugares *execute_task* e *finish_task* marcam, respectivamente, o início e o final da tarefa. Finalmente, o *release_resource* indica ao gerenciador que a tarefa foi encerrada e o recurso está novamente liberado.

O objetivo dos mecanismos aqui apresentados é facilitar a construção do nível de coordenação a partir do nível de *workflow*, pois uma vez definidas as interdependências entre as tarefas, a expansão ocorre utilizando o modelo da Figura 2 e os mecanismos de coordenação reutilizáveis.

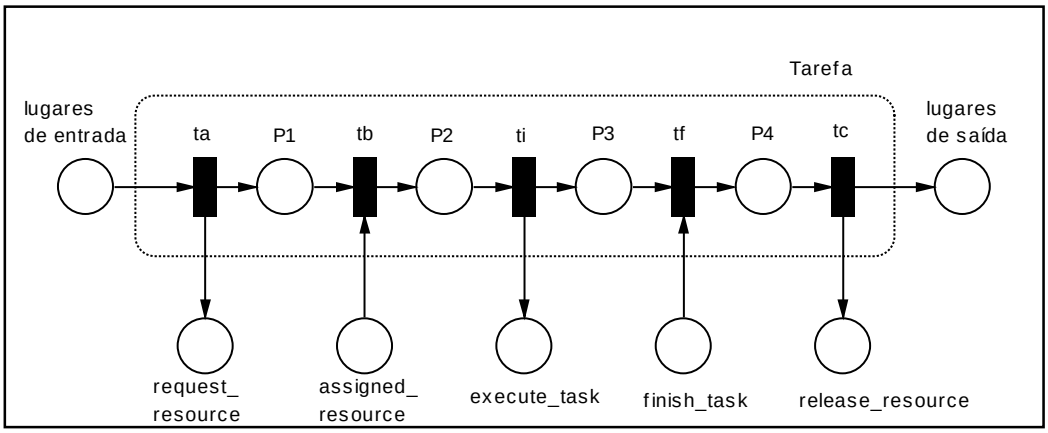

Figura 2: Estrutura de uma tarefa.

A partir desse modelo conclui-se que uma tarefa pode estar ligada a duas subredes, uma representando o gerenciador de recursos (que tem *request_resource* e *release_resource* como lugares de entrada e *assigned_resource* como lugar de saída) e outra representando a "lógica" da tarefa executada (tem *execute_task* como lugar de entrada e *finish_task* como lugar de saída). Estas duas subredes estão diretamente relacionadas às duas grandes classes de interdependências entre tarefas apresentadas nas próximas seções: dependências temporais e dependências de gerenciamento de recursos.

4.1. Dependências Temporais

As dependências temporais servem para estabelecer o ordenamento no processo de execução de tarefas. Os mecanismos propostos são baseados nas relações temporais definidas em um artigo clássico de lógica temporal [Allen 84], segundo o qual há um conjunto de relações primitivas e mutuamente exclusivas que podem ser aplicadas sobre intervalos de tempo. Estas relações entre intervalos de tempo foram adaptadas para a definição de dependências temporais entre tarefas, criando algumas variações das relações originalmente propostas e adicionando novas relações. As seguintes dependências temporais são definidas:

- *tarefa 1 equals tarefa 2*: esta dependência estabelece que duas tarefas devem ser executadas simultaneamente.
- *tarefa 1 starts tarefa 2*: a definição original de Allen estabelece que as tarefas devem começar juntas, e a tarefa 1 deve terminar antes (*startsA*). Retirando a restrição sobre qual tarefa deve terminar antes, criou-se uma nova dependência (*startsB*).
- *tarefa 1 finishes tarefa 2*: a definição original estabelece que as tarefas devem terminar juntas, mas a tarefa 1 deve começar depois da tarefa 2 (*finishesA*). É também possível eliminar a restrição sobre qual tarefa deve começar antes e criar uma nova relação (*finishesB*).
- *tarefa 2 after tarefa 1*: a tarefa 2 só pode ocorrer após a execução da tarefa 1. É possível definir duas variações para esta dependência. No primeiro caso (*afterA*), cada execução da tarefa 1 dá direito a uma execução da tarefa 2. No segundo caso (*afterB*), uma execução da tarefa 1 permite habilitar várias execuções da tarefa 2.
- *tarefa 1 before tarefa 2*: do ponto de vista da lógica temporal, esta relação pode ser vista como oposta à anterior, mas gera um mecanismo de coordenação totalmente diferente. Essencialmente, a restrição ocorre na execução da tarefa 1, que só pode ocorrer se a tarefa 2 ainda não tiver iniciado sua execução. A tarefa 2 não espera pela execução da tarefa 1, como ocorre na relação *tarefa 2 after tarefa 1*.

- *tarefa 1 meets tarefa 2*: a execução da tarefa 2 inicia imediatamente após o término da tarefa 1.
- *tarefa 1 overlaps tarefa 2*: a definição original estabelece que a tarefa 2 deve se iniciar antes do término da tarefa 1, que por sua vez deve terminar antes (*overlapsA*). Relaxando a restrição de que a tarefa 1 deve terminar antes, foi criada uma segunda relação (*overlapsB*).
- *tarefa 2 during tarefa 1*: possui duas variações. Na primeira delas (*duringA*), a tarefa 2 pode ser executada apenas uma vez durante a execução da tarefa 1. Na outra variação (*duringB*), a tarefa 2 pode ser executada várias vezes enquanto a tarefa 1 estiver sendo executada.

Para ilustrar o modelo dos mecanismos de coordenação para as dependências temporais, a Figura 3 mostra a PN que representa o mecanismo para a relação *tarefa 1 meets tarefa 2*. De acordo com esta relação, a tarefa 2 deve começar imediatamente após a tarefa 1. No nível de coordenação, esta relação é garantida bloqueando a conclusão da tarefa 1 enquanto a tarefa 2 não estiver pronta. No modelo da Figura 3, esta relação é satisfeita colocando o lugar *execute_task2* como entrada da transição que representa a tarefa 1 (transição não instantânea com reserva de *tokens*, como indicado pela letra "R" no desenho). Dessa forma, a tarefa 1 só será executada se a tarefa 2 estiver pronta para começar logo depois. Para manter a tarefa 2 habilitada, a tarefa 1 deve devolver o *token* ao lugar *execute_task2* após seu término. A conclusão da tarefa 1, habilita a tarefa 2 (*token* em *P1*).

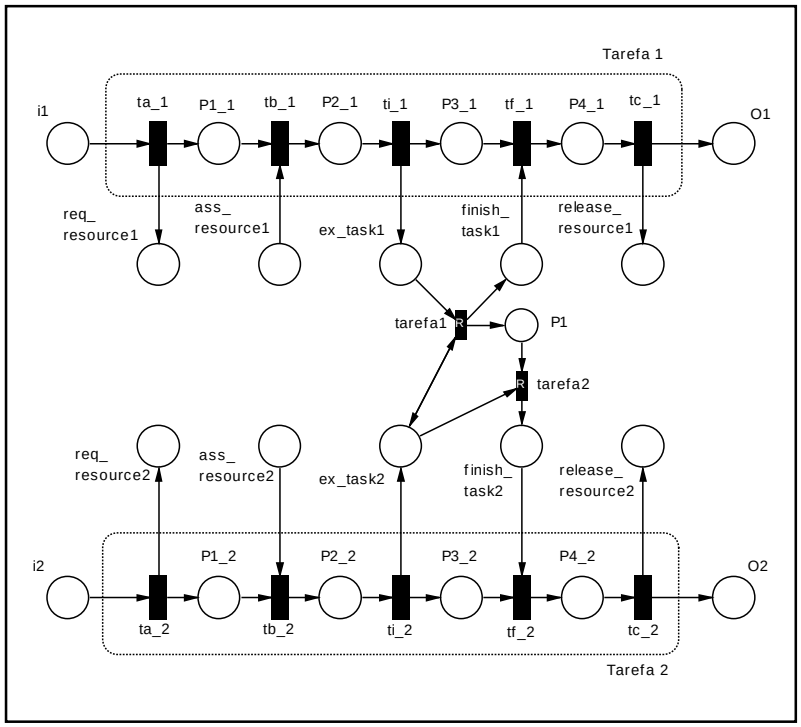

Figura 3: Mecanismo de coordenação para a dependência *tarefa 1 meets tarefa 2*.

Uma alternativa para evitar *deadlocks* é acrescentar mecanismos de *timeout*. Um tipo de *timeout* utilizado (*timeoutA*) define uma tarefa alternativa a ser executada se a original não se iniciar após um certo tempo de espera. Outro tipo de *timeout* (*timeoutB*) retorna *tokens* aos lugares de entrada da tarefa após um certo tempo de espera, de modo que a colaboração possa seguir um caminho alternativo que não passe pela tarefa bloqueada.

4.2. Dependências de Gerenciamento de Recursos

Os mecanismos de coordenação para o gerenciamento de recursos são complementares aos apresentados na seção anterior. Podem ser usados de forma independente, pois eles utilizam os lugares *request_resource*, *assigned_resource* e *release_resource* do modelo expandido de tarefa (Figura 2), não utilizados nos mecanismos da seção anterior.

Este tipo de mecanismo de coordenação lida com a distribuição dos recursos entre as tarefas. Há três mecanismos básicos:

- *Divisão de recursos*: um número limitado de recursos precisa ser compartilhado entre várias tarefas. É um caso muito comum que ocorre, por exemplo, quando vários computadores compartilham uma impressora, uma área de memória, etc.
- *Simultaneidade no uso de recursos*: o recurso só fica disponível se um determinado número de tarefas desejar utilizá-lo simultaneamente. É o caso de uma máquina que precisa de mais de um operador, por exemplo.
- *Volatilidade de recursos:* indica se após o uso, o recurso volta a estar disponível. A impressora não é um recurso volátil, mas uma folha de papel é.

Neste contexto, o termo "recurso" está sendo usado de maneira mais ampla que o normalmente usado em *workflows*, referindo-se não apenas ao agente que realiza a tarefa, mas também a qualquer artefato necessário à realização da tarefa.

As dependências de gerenciamento de recursos, diferentemente das temporais, não são relações binárias. É possível, por exemplo, que mais de duas tarefas compartilhem um recurso. Além disso, estas dependências requerem um parâmetro para indicar o número de recursos compartilhados, o número de tarefas que devem solicitar um recurso simultaneamente, ou o número de vezes que um recurso pode ser utilizado (volatilidade).

A Figura 4 ilustra o modelo para o gerenciador de recursos da divisão por 1 (duas tarefas compartilhando uma única instância do recurso – exclusão mútua). O modelo consiste em um lugar (*Pn*) com um *token*, representando o recurso. *Pn* serve de entrada para uma transição ligando *request_resource* a *assigned_resource*, definindo se há recursos disponíveis ou não para a execução da tarefa. Ao final da tarefa, uma transição saindo de *release_resource* devolve o *token* a *Pn*.

Também são definidos mecanismos compostos, a partir das três dependências básicas apresentadas acima, tais como *divisão por M com simultaneidade N* (até M grupos de N tarefas podem compartilhar o recurso) e *simultaneidade M com volatilidade N* (o recurso é disponibilizado para grupos de M tarefas, o que pode ser feito por até N vezes).

4.3. Implementação

Para automatizar a passagem do modelo no nível de *workflow* para o nível de coordenação, com a expansão das tarefas e a inserção dos mecanismos de coordenação, é proposto um esquema com três componentes: uma ferramenta de simulação de PNs, uma linguagem para a definição das dependências entre as tarefas e um programa capaz de criar uma nova PN para o nível de coordenação a partir da PN do nível de *workflow* e do arquivo com as dependências.

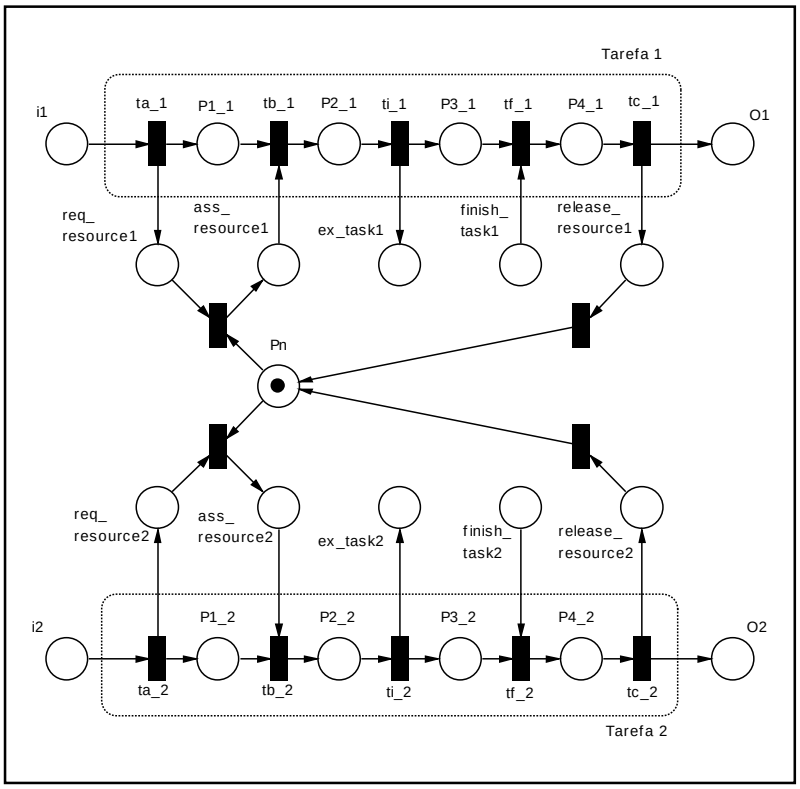

Figura 4: Mecanismo de coordenação para a exclusão mútua (divisão por 1).

A linguagem para a definição das dependências estabelece, uma a cada linha, todas as dependências existentes entre as tarefas de uma PN pré-modelada com a ferramenta de simulação. As dependências são definidas de acordo com a seguinte sintaxe `<nome da dependência> [parâmetros] "<nome da tarefa1>" "<nome da tarefa2>" ["<nomes das demais tarefas>"] [<timeouts>]`, onde os parâmetros são necessários para as dependências de gerenciamento de recursos, e os *timeouts* são opcionais.

A primeira implementação usa a ferramenta *freeware* para simulação de PNs Visual Simnet [Garbe 97], que possui editor gráfico para a modelagem das PNs, vários recursos para análise e um formato textual bastante simples para a definição das PNs que permite a exportação para outras ferramentas.

A partir do arquivo que define as dependências e do modelo inicial (nível de *workflow*) criado no Visual Simnet, foi implementado um programa para gerar uma nova PN no nível de coordenação, cujo comportamento pode ser analisado para a detecção de possíveis problemas nas interdependências entre as tarefas.

5. Exemplo

Para ilustrar o uso dos mecanismos de coordenação, é apresentado nesta seção um exemplo modelando uma situação de autoria colaborativa. O ambiente é composto por três usuários. Um deles é o dono do documento, que pode editá-lo, abri-lo e fechá-lo para a escrita dos demais autores. O usuário A é um usuário que pode ler o documento ou editá-lo, desde que o dono tenha liberado a escrita. O usuário B é um usuário especial, que sempre pode editar o documento, independentemente da autorização do dono.

Para este cenário, foram definidas três interdependências entre as tarefas. A primeira delas diz respeito à exclusão mútua na edição do documento (apenas um usuário pode editá-lo de cada vez). As outras duas dependências estão relacionadas à autorização para o usuário A editar o documento (ele só pode editar depois da liberação por parte do dono e antes que ele o feche para escrita). A representação do cenário descrito no nível de *workflow* é mostrada na Figura 5. Apesar deste cenário não tratar em detalhe o funcionamento de um sistema de autoria colaborativa, ele mostra como os mecanismos de coordenação podem ser utilizados em uma situação prática.

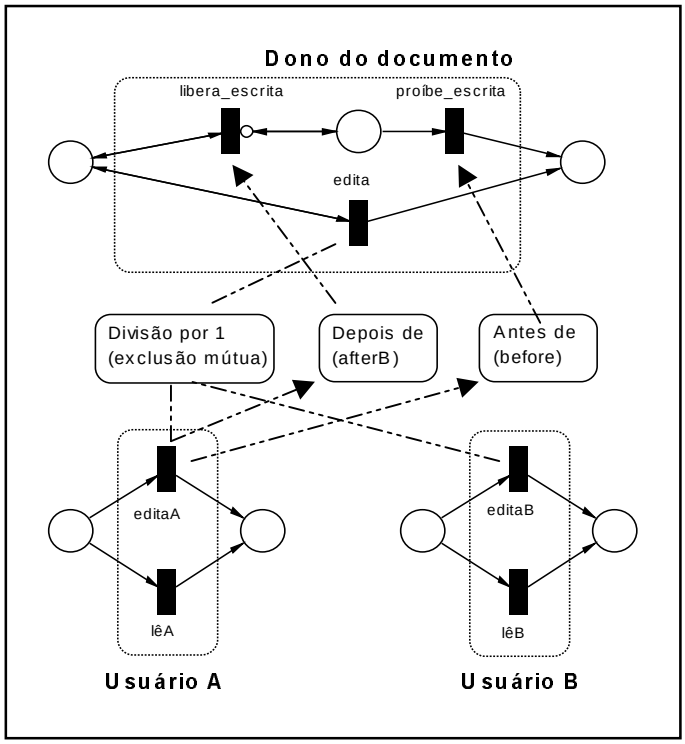

Figura 5: Exemplo modelado no nível de *workflow*.

A construção do nível de coordenação deste exemplo usando a aplicação apresentada na seção anterior é feita com o seguinte arquivo de dependências:

```
div 1 “edita” “editaA” “editaB”
afterB “editaA” “libera_escrita”
before “editaA” “proíbe_escrita”
```

O modelo no nível de coordenação é mostrado na Figura 6. Apesar de parecer complexa, a rede mostrada nesta figura é bastante modular e facilmente montada a partir da rede anterior (Figura 5) e dos mecanismos de coordenação pré-definidos. Com relação à expansão das tarefas, deve ser observado que nem todas as tarefas são expandidas exatamente como mostrado na Figura 2. O motivo é que esta expansão pode ser simplificada se uma tarefa possuir apenas dependências temporais ou de gerenciamento de recursos. Apenas as tarefas que possuírem os dois tipos de dependências são expandidas por completo (por exemplo, a tarefa “editaA”).

Para a PN do nível de coordenação, a análise de verificação indicou que não houve nenhuma situação de *deadlock*, exceto por dois estados finais corretos. A análise de validação comprovou que o sistema funciona como esperado: não ocorrem edições simultâneas e o usuário A não viola as autorizações do dono do arquivo. A análise de desempenho pode ser

realizada para medir, por exemplo, o tempo médio que um usuário espera para conseguir editar o documento, dadas as taxas com que cada usuário requisita o recurso de edição.

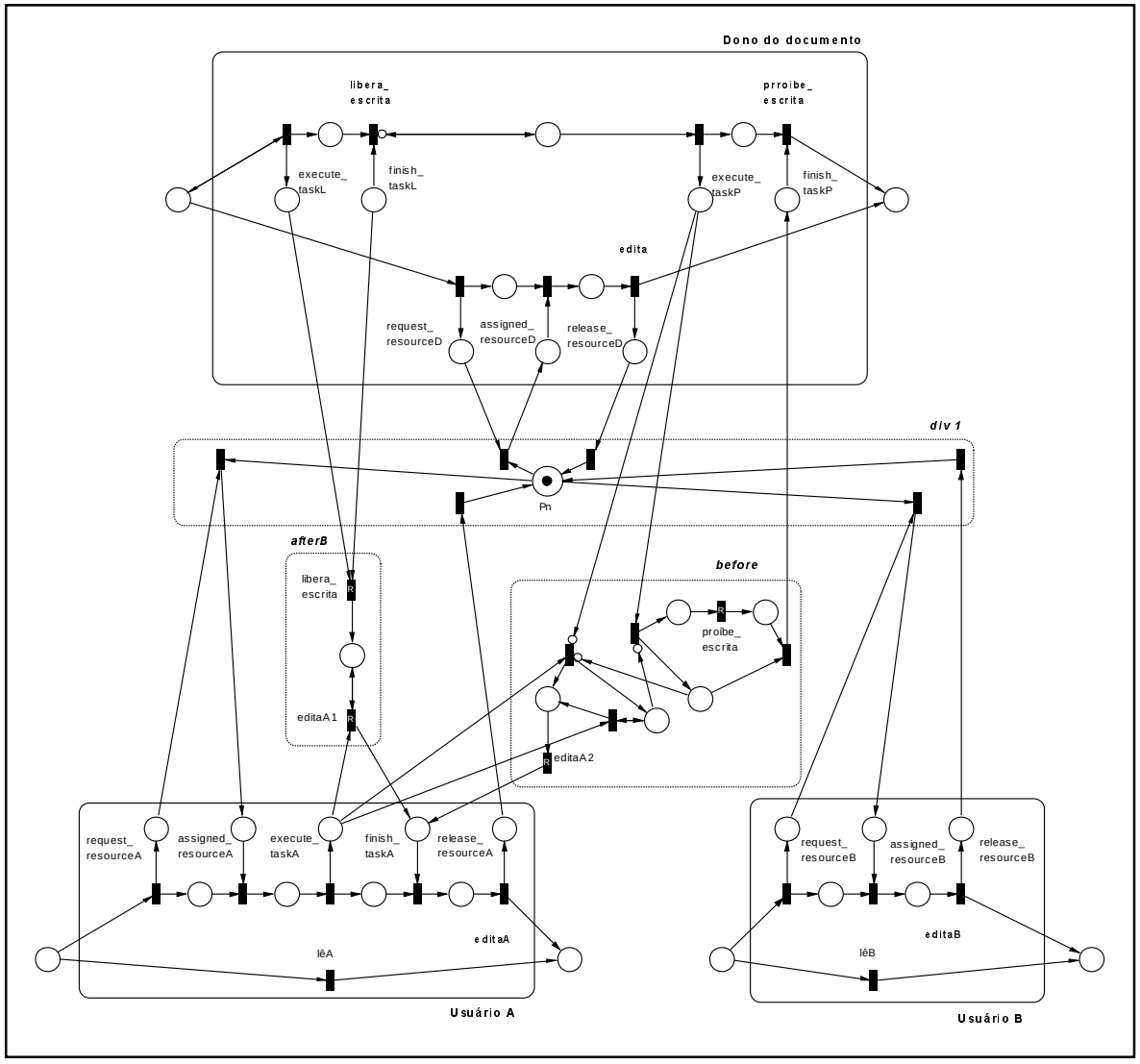

Figura 6: Exemplo modelado no nível de coordenação.

6. Conclusão

A coordenação de atividades interdependentes em ambientes colaborativos é um problema que deve ser tratado para garantir a eficiência da colaboração. A separação entre atividades e dependências, e a utilização de mecanismos de coordenação é uma maneira de lidar com este problema, que traz a vantagem da reutilização dos componentes em outras situações de colaboração. Além disso, a uniformidade do modelo facilita a padronização, desde que os elementos necessários sejam incorporados aos modelos de referência (por exemplo, [WfMC 98] e [WfMC 99]).

O conjunto de mecanismos de coordenação apresentado neste artigo não pretende abranger todas as possíveis dependências entre tarefas. A partir do uso, surgirão naturalmente novas dependências e mecanismos de coordenação que poderão ser agregados a este conjunto.

O uso de PNs para a modelagem dos mecanismos de coordenação se mostrou adequado tanto pela capacidade de análise e simulação das mesmas quanto pela sua descrição hierárquica, que permitiu definir a estrutura da coordenação em diferentes níveis de abstração (*workflow* e coordenação). No entanto, um problema típico de PNs é a explosão de estados, que pode ocorrer quando o número de tarefas interdependentes for muito grande. Atualmente, está sendo investigado como PNs de alto nível [Murata 89] podem simplificar os mecanismos de coordenação para minimizar este problema. Também pretende-se estudar a possibilidade de aplicação de metodologias como CPM (*Critical Path Method*) e PERT (*Program Evaluation and Review Technique*) [Nilsson 86] para concentrar o esforço de coordenação nas atividades do caminho crítico.

Devido à sua generalidade, os mecanismos propostos têm se mostrado adequados para uma série de sistemas colaborativos, desde *workflows* interorganizacionais [Raposo 00] (pretende-se disponibilizar estes mecanismos em sistemas de *workflow* atuais, tais como o Meteor [Meteor 99]) até ambientes virtuais colaborativos. Atualmente os mecanismos de coordenação estão sendo implementados para a utilização em ambientes virtuais colaborativos, onde usuários remotos estão simultaneamente presentes em um ambiente virtual 3D e podem interagir com objetos desse mundo e com os outros usuários [Benford 95]. A coordenação entre as atividades poderá permitir que este tipo de ambiente seja utilizado para a realização de tarefas colaborativas mais complexas que as atualmente realizadas, basicamente controladas pelo protocolo social.

Agradecimentos. O primeiro autor é bolsista de doutorado da FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), processo número 96/06288-0. Além da FAPESP, os autores agradecem ao DCA – FEEC – Unicamp pelo apoio expressivo dado à realização deste trabalho.

Referências

[Allen 84] J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23: 123-154. 1984.

[Bannon 91] L. J. Bannon and K. Schmidt. *CSCW: Four Characters in Search of a Context*. In J. M. Bowers and D. Benford (Eds.). *Computer Supported Cooperative Work*, pp. 3-16. North-Holland, 1991.

[Benford 95] S. Benford et al. Networked Virtual Reality and Cooperative Work. *Presence*, 4(4): 364-386. MIT Press, Fall 1995.

[De Cindio 88] F. De Cindio. G. De Michelis and C. Simone. *The Communication Disciplines of CHAOS*. In *Concurrency and Nets*, pp. 115-139. Springer-Verlag, 1988.

[Ellis 93] C. A. Ellis and G. J. Nutt. *Modeling and Enactment of Workflow Systems*. In M. A. Marsan (Ed.). *Application and Theory of Petri Nets 1993*, pp. 1-16. Lecture Notes in Computer Science 691. Springer-Verlag, 1993.

[Furuta 94] R. Furuta and P. D. Stotts. Interpreted Collaboration Protocols and their use in Groupware Prototyping. *Proc. of the Conf. on Computer-Supported Cooperative Work (CSCW'94)*, pp. 121-131. 1994.

[Garbe 97] W. Garbe. *Visual Simnet V.1.37 – Stochastic Petri-Net Simulator*. <http://home.arcor-online.de/wolf.garbe/simnet.html>. 1997.

[Gelernter 92] D. Gelernter and N. Carriero. Coordination Languages and their Significance. *Communications of the ACM*, 35(2): 97-107. February 1992.

[Holt 85] A. W. Holt. *Coordination Technology and Petri Nets*. In G. Rozenberg (Ed.). *Advances in Petri Nets*, pp. 278-296. Lecture Notes in Computer Science, 222. Springer-Verlag, 1985.

[Li 98] D. Li and R. Muntz. COCA: Collaborative Objects Coordination Architecture. *Proc. of the Conf. on Computer-Supported Cooperative Work (CSCW'98)*, pp. 179-188. 1998.

[Magalhães 98] L. P. Magalhães, A. B. Raposo and I. L. M. Ricarte. Animation Modeling with Petri Nets. *Computer & Graphics*, 22(6): 735-743. Pergamon Press, 1998.

[Malone 90] T. W. Malone and K. Crowston. What Is Coordination Theory and How Can It Help Design Cooperative Work Systems? *Proc. of the Conf. on Computer-Supported Cooperative Work (CSCW'90)*, pp. 371-380. 1990.

[Malone 95] T. W. Malone, K-Y. Lai and C. Fry. Experiments with Oval: A Radically Tailorable Tool for Cooperative Work. *ACM Transaction on Information Systems*, 13(2): 177-205. April 1995.

[Merz 97] M. Merz, B. Liberman and W. Lamersdorf. Using Mobile Agents to support Interorganizational Workflow Management. *International Journal of Applied Artificial Intelligence*, 11(6). September 1997.

[Meteor 99] LSDIS – Dept. of Computer Science – Univ. of Georgia. *METEOR$_2$ (Managing End-To-End OpeRations)*. <http://lsdis.cs.uga.edu/proj/meteor/meteor.html>

[Murata 89] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4): 541-580. April 1989.

[Nilsson 86] N. J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann Pubs, 1986.

[Raposo 00] A. B. Raposo, L. P. Magalhães and I. L. M. Ricarte. Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. Aceito para o *Int. J. of Computer Systems Science & Engineering, Special Issue on Flexible Workflow Technology Driving the Networked Economy*. CRL Publishing, September 2000.

[van der Aalst 94] W. M. P. van der Aalst, K. M. van Hee and G. J. Houben. Modelling and analysing workflow using a Petri-net based approach. *Proc. of the 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pp. 31-50. 1994.

[van der Aalst 98] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1): 21-66. 1998.

[WfMC 96] Workflow Management Coalition. *Interface 4 – Interoperability Abstract Specification*, WFMC-TC-1012 (October 1996). <http://www.aiim.org/wfmc/standards/docs/if4-a.pdf>

[WfMC 98] Workflow Management Coalition. *Interface 5 – Audit Data Specification – Version 1.1*, WFMC-TC-1015 (September 1998). <http://www.aiim.org/wfmc/standards/docs/if4v11b.pdf>

[WfMC 99] Workflow Management Coalition. *Interface 1 – Process Definition Interchange Process Model – Version 1.1*, WFMC-TC-1016-P (October 1999). <http://www.aiim.org/wfmc/standards/docs/if19910v11.pdf>

[Winograd 86] T. Winograd and F. Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Ablex, 1986.