# Fluxo de dados em redes de Petri coloridas e em grafos orientados a atores

Grace Anne Pontes Borges

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. João Eduardo Ferreira

- São Paulo, outubro de 2008 -

# Fluxo de dados em redes de Petri coloridas e em grafos orientados a atores

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Grace Anne Pontes Borges e aprovada pela Comissão Julgadora.

São Paulo, 09 de outubro de 2008.

Banca Examinadora:

• Prof. Dr. João Eduardo Ferreira (orientador) - IME-USP

• Prof. Dr. Roberto Hirata Junior - IME/USP

• Prof. Dr. Jorge Rady de Almeida Junior - EP/USP

*Ao meu marido José Paulo*
*por sua compreensão, incentivo e*
*companheirismo na concretização*
*deste trabalho.*

# Agradecimentos

Ao meu orientador, Prof. Dr. João Eduardo Ferreira, pelos ensinamentos, atenção e apoio. Seu exemplo de excelente professor e profissional despertou meu interesse pelo mestrado. Obrigada pela oportunidade, confiança no meu trabalho e paciência nos momentos difíceis.

Agradeço especialmente à minha mãe por tudo que me ensinou ainda em vida e por tudo que ainda aprendo com ela depois de sua partida. Sei que, onde estiver, sempre torcerá por mim. Obrigada D. Neila.

Ao imenso carinho, apoio e dedicação do meu pai Herivelto e sua esposa Joelma. Aos meus irmãos, por todo incentivo nessa caminhada, pelas palavras de conforto nos momentos difíceis, pelos conselhos, orientações e contribuições para este trabalho, cada um à sua maneira, provando que distância física não é problema quando queremos estar presentes na vida de quem amamos.

Ao meu marido José Paulo, grande incentivador deste e de todos meus projetos de vida, por tudo o que compartilhamos diariamente, inclusive sua família, que também tem sido a minha.

Aos professores Cristina Gomes Fernandes e Roberto Hirata Jr. pelos comentários feitos em meu exame de qualificação, contribuindo para o enriquecimento desta dissertação.

Aos amigos do IME-USP, Ana Beatriz Graciano, Arnaldo Câmara Lara, David da Silva Pires, Márcio Katsumi Oikawa e William Tankiti Yamamoto pelo companheirismo e pelas contribuições diretas ou indiretas a este mestrado. Em especial, agradeço a Kelly Rosa Braghetto e Pedro Losco Takecian por todo apoio dado, pelo que me ensinaram, por estarem ao meu lado em momentos difíceis, compartilhando também os momentos bons e por toda a ajuda na realização deste trabalho.

Aos colegas de trabalho e amigos professores da FATEC-SP, por acreditarem no meu trabalho e me incentivarem cada vez mais na carreira acadêmica.

Aos meus amigos da turma FATEC-PD/97 e “agregados”, pela força, carinho e momentos de descontração.

Aos amigos da Associação ARACÊ pelos vários momentos de reflexão e compreensão sobre a importância deste trabalho na minha vida, cada pensamento positivo, cada demonstração de carinho, especialmente a Ana Seno e Marcelo Rouanet, que, além de amigos muito queridos, contribuíram com a revisão deste texto.

A todos que contribuíram e acompanharam este trabalho de diferentes maneiras.

# Resumo

Há três décadas, os sistemas de informação corporativos eram projetados para apoiar a execução de tarefas pontuais. Atualmente, esses sistemas também precisam gerenciar os fluxos de trabalho (*workflow*s) e processos de negócio de uma organização. Em comunidades científicas de físicos, astrônomos, biólogos, geólogos, entre outras, seus sistemas de informações distinguem-se dos existentes em ambientes corporativos por: tarefas repetitivas (como re-execução de um mesmo experimento), processamento de dados brutos em resultados adequados para publicação; e controle de condução de experimentos em diferentes ambientes de *hardware* e *software*.

As diferentes características dos dois ambientes – corporativo e científico – propiciam que ferramentas e formalismos existentes ou priorizem o controle de fluxo de tarefas, ou o controle de fluxo de dados. Entretanto, há situações em que é preciso atender simultaneamente ao controle de transferência de dados e ao controle de fluxo de tarefas.

Este trabalho visa caracterizar e delimitar o controle e representação do fluxo de dados em processos de negócios e *workflow*s científicos. Para isso, são comparadas as ferramentas *CPN Tools* e KEPLER, que estão fundamentadas em dois formalismos: redes de Petri coloridas e grafos de *workflow* orientados a atores, respectivamente. A comparação é feita por meio de implementações de casos práticos, usando os padrões de controle de dados como base de comparação entre as ferramentas.

Palavras-chave: redes de Petri coloridas, CPN Tools, *workflows* científicos, KEPLER, padrões de fluxo de dados, grafos de workflow orientados a atores

# Abstract

Three decades ago, business information systems were designed to support the execution of individual tasks. Today's information systems also need to support the organizational workflows and business processes. In scientific communities composed by physicists, astronomers, biologists, geologists, among others, information systems have different characteristics from those existing in business environments, like: repetitive procedures (such as re-execution of an experiment), transforming raw data into publishable results; and coordinating the execution of experiments in several different software and hardware environments.

The different characteristics of business and scientific environments propitiate the existence of tools and formalisms that emphasize control-flow or dataflow. However, there are situations where we must simultaneously handle the data transfer and control-flow.

This work aims to characterize and define the dataflow representation and control in business processes and scientific workflows. In order to achieve this, two tools are being compared: CPN Tools and KEPLER, which are based in the formalisms: colored Petri nets and actors-oriented workflow graphs, respectively. The comparison will be done through implementation of practical cases, using the dataflow patterns as comparison basis.



*Keywords*: colored Petri nets, CPN Tools, scientific workflows, KEPLER, dataflow patterns, actors-oriented workflow graphs

# Sumário

# Lista de Figuras

# Lista de Tabelas

**Capítulo 1**

# Introdução

## 1.1. Motivação

Tradicionalmente, sistemas de informação corporativos eram projetados para apoiar a execução de tarefas pontuais. Atualmente, esses sistemas de informação também precisam controlar, monitorar e apoiar o gerenciamento de processos de negócio e fluxos de trabalho (*workflows)* [3].

Em comunidades científicas de físicos, astrônomos, biólogos, geólogos, entre outras, os sistemas de informações diferem dos existentes em ambientes corporativos nos seguintes aspectos: realizam tarefas repetitivas como re-execução de um mesmo experimento; transformam dados brutos em resultados adequados para publicação; e controlam experimentos em diferentes ambientes de *hardware* e *software*. Para diferenciar *workflows* utilizados em ambientes corporativos dos científicos foi criado o termo *workflows científicos* [21].

Ao analisarmos projetos e modelos de execução no gerenciamento de processos de negócios, a prioridade é o controle de fluxo das tarefas, enquanto o fluxo de dados propriamente dito geralmente fica em segundo plano. Por outro lado, sistemas de *workflows* científicos tendem à execução mais orientada a fluxo de dados, pois gerenciam aplicações com grande volume de dados, normalmente heterogêneos e distribuídos [22]. Essas diferenças incentivam a busca por alternativas que tratem fluxo de tarefas em conjunto com fluxo de dados ampliando a capacidade de representação de *workflows* em aplicações científicas e em ambientes de negócios.

Tanto os processos de negócio, quanto os *workflows* científicos podem ser representados por meio de modelos, que são descrições de processos suficientemente detalhadas de modo que possam ser executadas diretamente por um sistema de gerenciamento de workflow. Algumas das linguagens de modelagem existentes apresentam arcabouços formais que permitem análises dos processos, com objetivo de detecção de falhas e possíveis melhorias no projeto. Existem formalismos que priorizam a representação do fluxo de tarefas, por exemplo, as álgebras de processos [10]. Outros formalismos apresentam estruturas que permitem também a representação de dados, como as redes de Petri e suas extensões [9] e os grafos de *workflow* [22].

## 1.2. Objetivo

Este trabalho tem como objetivo caracterizar e delimitar o controle e a representação de dados em duas ferramentas: *CPN Tools* e sistema KEPLER. A primeira é usada para editar, simular e analisar redes de Petri coloridas (Seção 2.1), uma abordagem amplamente utilizada para representação de processos de negócios. E a segunda tem sido bastante usada para projetar, executar e monitorar *workflows* científicos e está fundamentada em grafos de *workflow* orientados a atores (Seção 2.2).

Busca-se identificar a abrangência e os limites de cada ferramenta, comparando casos práticos. Para tal finalidade, analisou-se o poder de representação dos padrões de controle de dados definidos por Russel *et al* [1] em ambas ferramentas.

## 1.3. Contribuições

As principais contribuições deste trabalho são:

- Caracterização dos limites das redes de Petri coloridas para representação de padrões de dados por meio da ferramenta *CPN Tools*;
- Caracterização dos limites de grafos de *workflow* orientado a atores para representação de padrões de dados por meio do sistema KEPLER;
- Desenvolvimento da representação dos padrões de controle de dados de *workflow* em redes de Petri coloridas e grafos de *workflow* orientado a atores;
- Comparação entre implementações em *CPN Tools* e KEPLER utilizando os padrões de controle de dados de *workflow*.

## 1.4. Organização do texto

O restante do texto está organizado da seguinte forma: o Capítulo 2 apresenta fundamentos teóricos básicos para compreensão deste trabalho, como redes de Petri e sua extensão, redes de Petri coloridas (rdPc), além de *workflows* científicos e grafos de *workflow*. O Capítulo 3 apresenta a ferramenta *CPN Tools* e o sistema KEPLER que serão usados para implementação dos casos práticos. No Capítulo 4 serão apresentados os padrões de controle de dados e a comparação entre redes de Petri coloridas e *workflows* científicos usando as ferramentas apresentadas no capítulo anterior. No Capítulo 5, é apresentada a conclusão do trabalho.

**Capítulo 2**

# Fundamentos

Este capítulo apresenta os formalismos que embasam as ferramentas comparadas neste trabalho. Porém, antes de aprofundar esses fundamentos, é importante apresentar um conjunto de definições geralmente utilizadas nos vários trabalhos sobre *workflows*, inclusive neste.

## 2.1. Conceitos de *Workflow*

Segundo a WfMC (*Workflow Management Coalition*) [7], workflow é a automação, completa ou parcial, de um processo de negócio, na qual documentos, informações ou tarefas são passados de um participante para outro por meio de ações seguindo um conjunto de regras de procedimento. Um sistema de gerenciamento de workflow define, cria e gerencia a execução de workflows usando software.

Modelo de *workflow* é a descrição de um processo de negócios suficientemente detalhada de modo que o processo possa ser executado diretamente por um sistema de gerenciamento de *workflow* [1]. Um modelo pode ser representado por uma série de tarefas ligadas na forma de grafo dirigido. Os elementos básicos de um *workflow* são ilustrados na Figura 2.1 e descritos a seguir.

- *Caso* ou *instância de processo*: instância de execução de um modelo de *workflow*;
- *Tarefa*: corresponde a uma unidade de trabalho. Pode ser:

  a) *Atômica*: possui uma definição simples e autocontida, não descrita em termos de outras tarefas. Apenas uma instância da tarefa executa quando iniciada;

  b) *Bloco de tarefas*: ação complexa com implementação descrita em termos de *subworkflow*. Quando um bloco de tarefas é iniciado, ele passa o controle para a primeira tarefa do seu *subworkflow* correspondente. O *subworkflow* realiza todas as tarefas que o compõem devolvendo

então o controle ao bloco de tarefas após sua conclusão. No exemplo da Figura 2.1, o bloco de tarefas C define-se em termos do *subworkflow* compreendendo as tarefas X, Y e Z;

c) *Instância múltipla*: uma tarefa que pode ter múltiplas instâncias distintas executando simultaneamente no mesmo caso de *workflow*;

d) *Instância múltipla de bloco de tarefas*: denota um bloco de tarefas que pode ter múltiplas instâncias distintas de execução;

- *Instância de tarefa*: cada tarefa em execução;

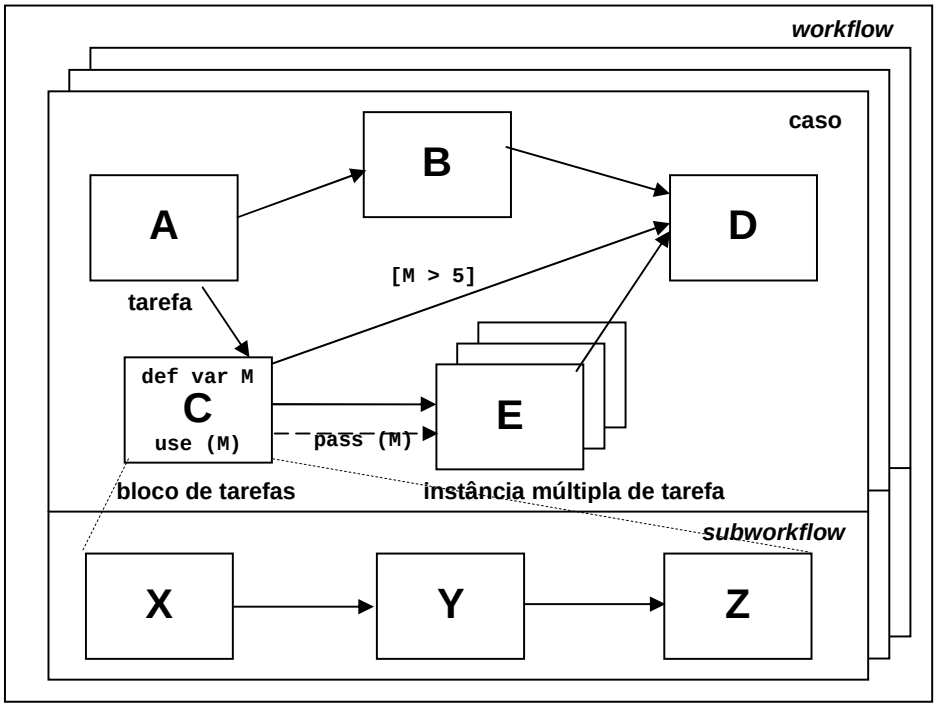



Figura 2.1. Componentes de um *workflow* [1]

- *Canal de controle*: meio pelo qual ocorre o controle de fluxo das tarefas, ou seja, a seqüência em que as tarefas serão realizadas. Representado na Figura 2.1 pelas setas cheias (→);
- *Canal de dados*: meio de comunicação de dados entre duas tarefas. Representado na Figura 2.1 pelas setas pontilhadas;
- *Dados*: elementos usados pelas tarefas para armazenar informações de produção, ou pelo *workflow* para gerir controle de fluxo de tarefas. Também podem ser elementos de comunicação com o

ambiente externo. São representados por variáveis e podem ser passados de uma tarefa para outra por meio dos canais de dados. Na Figura 2.1, a variável *M* é definida no escopo da tarefa *C* para armazenamento de dado local. Esse mesmo valor enviado à tarefa E também é usado para gerenciar o controle de fluxo para a tarefa *D* (*M>5*).

## 2.2. Redes de Petri

O termo redes de Petri denomina uma técnica de especificação formal largamente difundida e adequada para modelagem de sistemas com atividades paralelas, concorrentes, assíncronas e não determinísticas [9].

Redes de Petri são utilizadas: na área de protocolo de comunicação, em sistemas operacionais, no projeto de *hardware*, em sistemas embutidos, no projeto de sistemas de *software* e reengenharia de processos de negócios. Sua teoria foi apresentada em 1962 por Carl Adam Petri em sua tese de doutorado.

**Definição 2.2.1:** Uma *rede de Petri* clássica é dada por uma quíntupla **RP = (P, T, A, W, $M_0$)** em que:

(i) (**P,T,A**) é um grafo bipartido dirigido [1], com conjunto de vértices P∪T e conjunto de arcos A;

(ii) **W** é uma função, chamada *peso*, que associa um número natural não nulo a cada arco de A, ou seja, **W: A →** {1, 2, 3, ...};

(iii) **$M_0$** é uma função, chamada *marcação inicial*, que associa um número natural a cada vértice de P, ou seja, **$M_0$: P →** {0, 1, 2, ...};

Os vértices de T denominam-se *transições* e representam as tarefas realizadas pelo sistema modelado, enquanto os vértices de P são chamados de *lugares* ou *estados*, representando variáveis de estado do sistema. Quando o arco parte de um lugar $p \in$ P e incide em uma transição $t \in$ T, *p* é chamado *lugar de entrada* de *t* e representa uma pré-condição a ser atingida para que *t* ocorra. Quando um arco parte de uma transição *t* e incide em um lugar *p*, este é chamado *lugar de saída* de *t* e representa um estado do sistema atingido após ocorrência de *t*.

O funcionamento de uma rede de Petri é controlado por meio de *fichas* (*tokens*) dispostas nos lugares e também pelo *peso* associado aos seus arcos. A função $M_0$ dá o posicionamento inicial das fichas. Para cada vértice *p* em P, considera-se que há $M_0$(p) fichas em *p*. Em outras palavras, $M_0$(p) representa o número inicial de fichas em *p*. A função W(a) indica a quantidade de fichas a serem consumidas do lugar de entrada vinculado ao

1 Um *grafo bipartido dirigido* (V, A) é aquele em que o conjunto de vértices V é dividido em dois conjuntos disjuntos P e T, tal que todo arco de A tem uma ponta em P e outra em T e será denotado por (P, T, A).

arco *a* ou a quantidade de fichas a serem acrescidas no lugar de saída vinculado a esse arco. Quando o peso associado ao arco é 1, sua representação na rede pode ser omitida.

Apresenta-se a seguir um exemplo de redes de Petri para descrever seu funcionamento, conforme Figura 2.2. Utiliza-se um "processo de atendimento a reclamações" no qual é possível identificar: as transições (*registrar*, *pagar e enviar carta*); os lugares (*Reclamação*, *Sob avaliação* e *Pronto*); os arcos de peso 1 e ainda marcas iniciais (no lugar *Reclamação*), representando três reclamações a serem atendidas.

Uma transição *t* está *habilitada* se, e somente se, para todo lugar de entrada da transição, o lugar possuir um número de marcas superior ou igual ao peso do arco ligando o lugar à transição. Essa regra habilita o disparo de uma transição, mas não é obrigatória (não-determinismo).

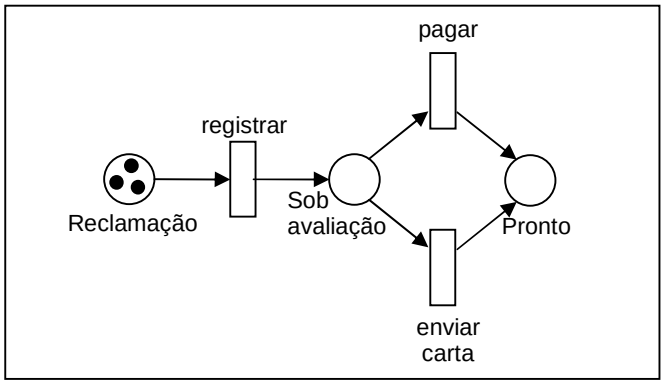



Figura 2.2. Rede de Petri para atendimento a reclamações

O *disparo* de uma transição subtrai marcas de todos os seus lugares de entrada (segundo o peso dos arcos ligando os lugares de entrada à transição), acrescentando marcas em todos os seus lugares de saída (segundo o peso dos arcos ligando a transição aos seus lugares de saída).

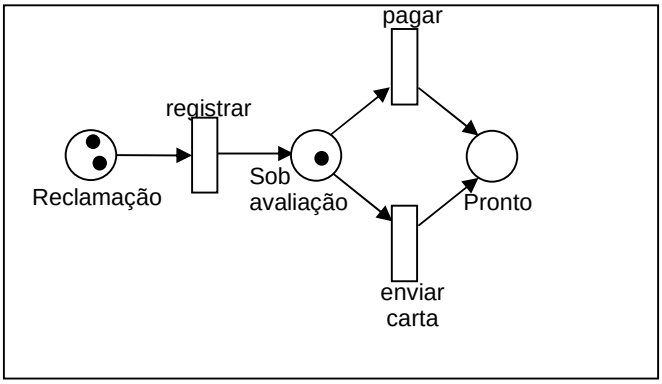



Figura 2.3. Disparo da transição *registrar*

A marcação de uma rede de Petri muda a cada disparo de transição, permitindo simular o comportamento dinâmico do sistema e definir seu estado em dado momento. O exemplo da Figura 2.3 mostra o disparo da transição *registrar*, consumindo uma marca do lugar de entrada *Reclamação* e acrescentando uma marca no lugar de saída *Sob avaliação*.

### 2.2.1. Redes de Petri coloridas (rdPc)

Nas redes de Petri clássicas existe apenas um tipo de marca, não permitindo diferenciar recursos em um lugar. Ademais, as redes de Petri clássicas não são hierárquicas, portanto, não permitem a construção de modelos a partir de submodelos separados com interfaces bem definidas [11].

Entre os anos 70 e 80, as redes de Petri foram estendidas considerando conceito de dados e hierarquia, originando, assim, as redes de Petri coloridas e hierárquicas [3]. As redes de Petri coloridas (rdPc) incorporam tipos de dados (conjuntos de cores) para representação dos dados e podem utilizar a decomposição hierárquica, sem perder a consistência das mesmas em relação às originais.

As rdPc se mostram um formalismo adequado para aprofundar e entender o controle de dados em processos de negócios, devido à possibilidade de representação de dados, o que não ocorre com redes de Petri clássicas [14].

As redes de Petri coloridas combinam a expressividade das redes de Petri com a expressividade das linguagens de programação. Redes de Petri oferecem primitivas para descrever sincronização de processos concorrentes, enquanto linguagens de programação oferecem primitivas para definição de tipos de dados (conjuntos de cores) e manipulação de dados [16].

A principal finalidade das redes de Petri coloridas é a redução do tamanho do modelo, permitindo que marcas distintas (de cores diferentes) representem diferentes processos ou recursos na mesma sub-rede.

### 2.2.2. Definição formal de redes de Petri coloridas

Para a definição formal de redes de Petri coloridas utilizar-se-á dois conceitos importantes: *multiconjunto* e *expressões*.

**Definição 2.2.2**: *Multiconjunto* é um objeto cuja representação se assemelha à de um conjunto finito, mas diferindo deste por permitir a existência do mesmo elemento várias vezes. Supondo-se um conjunto {b, c, f}, ao se adicionar o elemento *c*, ainda tem-se o conjunto {b, c, f}. Entretanto, supondo-se o multiconjunto {b, c, f}, ao se adicionar o elemento *c*, o resultado é o multiconjunto {b, c, c, f}.

Um multiconjunto é sempre definido sobre um conjunto S, ou seja, seus elementos pertencem a S. Um multiconjunto pode ser descrito como uma soma formal em que cada elemento de S apresenta um coeficiente dizendo quantas vezes esse elemento aparece no multiconjunto. Por convenção, elementos com coeficiente zero são omitidos. Por exemplo, dado o multiconjunto m = {a, c, c, e}, definido sobre S = {a, b, c, d, e}, o mesmo pode ser representado pela soma m = 1´a + 2´c + 1´e.

**Definição 2.2.3**: *Expressões* são estruturas com constantes, variáveis e operadores, neste contexto, representam funções que associam valores de um dado domínio a valores booleanos ou a multiconjuntos. Chamar-se-á *expressões fechadas* as funções constantes, ou seja, compreendendo apenas constantes e operadores, sem variáveis.

**Definição 2.2.4:** As redes de Petri coloridas são definas pela tupla rdPc = (Σ, P, T, A, C, G, E, I), em que:

(i) **Σ** é um conjunto finito não-vazio de tipos (ou *conjuntos de cores*), ou seja, é um conjunto de conjuntos de cores;

(ii) (**P,T,A**) é um grafo bipartido dirigido, com conjunto de vértices P∪T e conjunto de arcos A;

(iii) **C** é uma função, chamada *cor***,** que associa um tipo (conjunto de cores) definido em Σ a cada vértice de **P**, ou seja, **C: P → Σ**.

(iv) **B** é um conjunto de *expressões de tipo booleano* representadas por funções **b : Σ → {V, F}** e **G** é uma função, chamada *expressão de guarda***,** que associa um elemento de **B** a cada vértice de **T**, ou seja, **G: T → B**.

(v) **E** é uma função, chamada *expressão de arco***,** que associa uma expressão com domínio em Σ a cada arco de **A**. A imagem de cada expressão de arco deve ser um multiconjunto definido sobre o tipo associado a *p*(a), em outras palavras, um multiconjunto com elementos da mesma cor associada ao vértice *p* que está vinculado ao arco *a*, ou seja, **E: A → {expressões}**;

(vi) **I** é uma função, chamada *inicialização*, que associa uma expressão fechada a cada vértice de **P**. A imagem de cada expressão de inicialização deve ser um multiconjunto definido sobre o tipo associado ao vértice *p*.

Os vértices de P são chamados de *lugares* ou *estados*, enquanto os vértices de T são chamados de *transições*. Além de definir tipos, o conjunto **Σ** determina as operações, funções e variáveis que podem ser usadas nas inscrições da rede, ou seja, determina as *declarações* necessárias a cada modelo.

O funcionamento de uma rede de Petri é controlado por multiconjuntos de *fichas* (*tokens*) dispostas nos lugares e também por *inscrições* associadas à estrutura da rede, como por exemplo: expressões de arco, expressões de guarda, tipos de cores associados aos lugares e expressões de inicialização.

Cada multiconjunto de fichas dispostas em um lugar *p* deve ser definido sobre o tipo associado a *p*, ou seja, cada ficha deve apresentar uma cor que pertença ao tipo C(*p*). O mesmo cuidado deve ser tomado com expressões de arco, pois determinam a quantidade e tipo (multiconjunto) de fichas a serem consumidas ou depositadas em lugar vinculado ao arco em questão.

Cada expressão de guarda associada a uma transição $t \in$ T é usada como pré-condição a ser atendida dentro do modelo representado, habilitando ou não *t* para disparo, ou seja, define se tarefa associada *t* pode ser realizada.

A função I dá o posicionamento inicial das fichas. Para cada $p \in$ P, considera-se que há I(p) fichas em *p*. Em outras palavras, I(p) representa o multiconjunto inicial de fichas em *p*.

A Figura 2.4 apresenta um exemplo de rede de Petri colorida. Utilizar-se-á um “processo de atendimento a reclamações”, que após o atendimento, computa clientes pagos e cartas enviadas e reporta ao responsável pelo processo, exemplificando os diferentes tipos de fichas que podem existir numa rede de Petri colorida. É possível identificar:

- Conjunto de cores com a definição dos tipos: R e INT com seus respectivos elementos e a declaração de variáveis (ce, cp e tr) do tipo definido INT;
- Transições: registrar, pagar, enviar carta, reportar;
- Lugares: Reclamação, Sob avaliação, Carta enviada, Cliente pago e Pronto;
- Arcos dirigidos entre transições e lugares;
- Cor R associada aos lugares: Reclamação e Sob avaliação, e cor INT associada aos lugares: Carta enviada, Cliente pago e Pronto;
- Expressão de guarda [ce+cp>3] associada à transição *reportar*, representando pré-condição para o disparo dessa transição. A tarefa *reportar* só pode ser realizada caso a quantidade de cartas enviadas mais a quantidade de clientes pagos exceda três;
- Uma expressão de arco associada a cada arco da rede. Algumas possuem expressões fechadas (1`r e 1`0), enquanto outras utilizam expressões compostas por variáveis do tipo associado ao lugar vinculado ao arco (ce, ce+1, cp, cp+1, tr, tr+ce+cp);

- Também é definida a inicialização da rede, ou seja, quantidade e tipo de marcas apresentadas inicialmente em cada lugar. O valor inicial do lugar Reclamação é 10`r(dez fichas de valor r), que representa dez reclamações a serem atendidas.

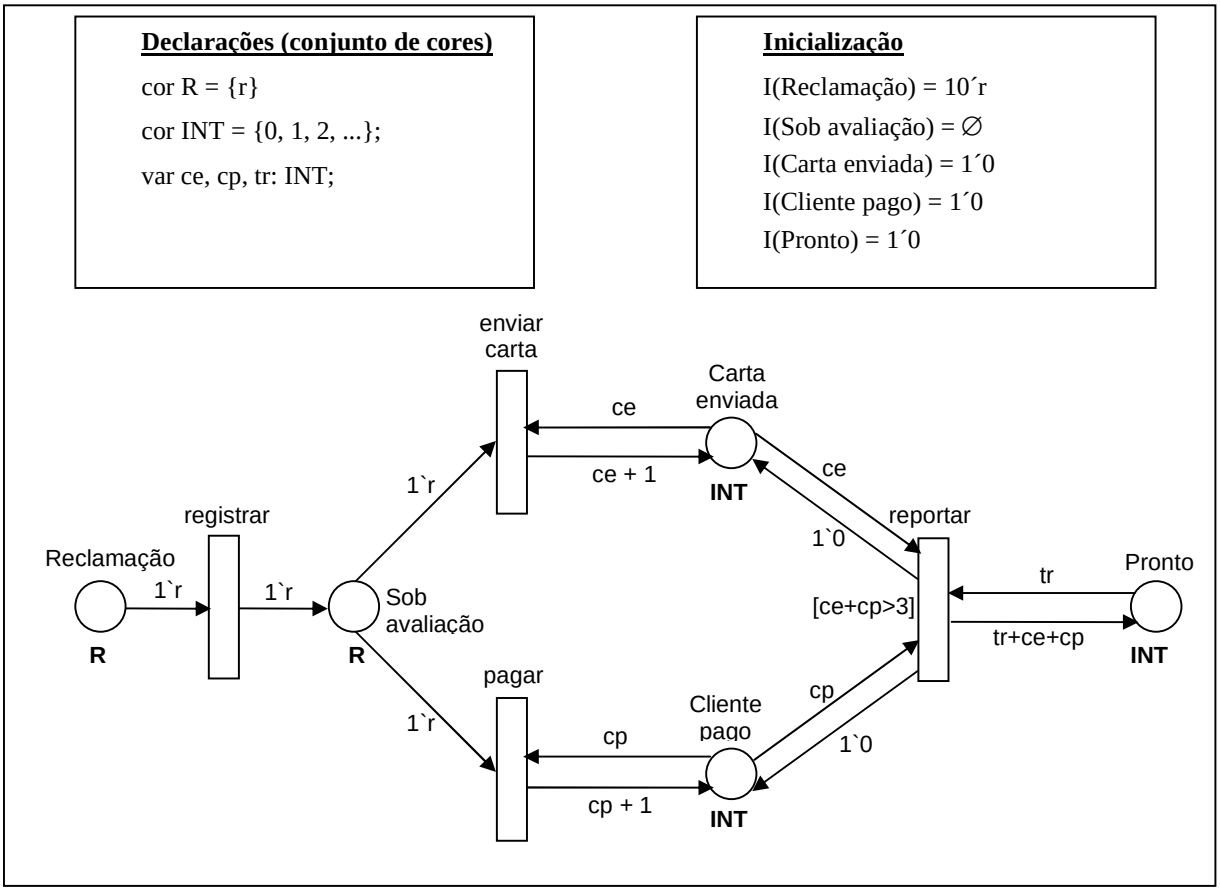



Figura 2.4. Rede de Petri colorida para atendimento a reclamações

De maneira semelhante à rede de Petri clássica, uma transição *t* está *habilitada* se, e somente se, para todo lugar de entrada de *t*, o lugar possuir um multiconjunto de fichas satisfazendo à expressão associada ao arco que liga o lugar à transição. Caso a transição possua expressão de guarda associada, esta também precisa ser satisfeita para que *t* seja habilitada.

O disparo de uma transição subtrai fichas de todos os seus lugares de entrada (segundo as expressões associadas aos arcos ligando os lugares de entrada à transição) e deposita fichas em todos os seus lugares de saída (segundo as expressões associadas aos arcos ligando a transição aos seus lugares de saída).

## 2.3. *Workflows* científicos

Apesar das definições relacionadas a *workflow* se referirem a processos de negócios, os sistemas de gerenciamento de *workflow* vêm sendo usados também em aplicações científicas.

Porém, o ambiente científico difere do ambiente de negócios: re-executa um mesmo experimento, controla condução de experimentos em diferentes ambientes de *hardware* e *software*, trabalha com grande volume de dados, normalmente heterogêneos e distribuídos. Devido a essas peculiaridades, os *workflows* desse ambiente são chamados *workflows* científicos.

As características e requisitos de *workflows* científicos assemelham-se parcialmente as de *workflow* de negócios. Historicamente, o *workflow* de negócios tem sua origem em sistemas de automação de escritório e, mais recentemente, teve seu uso destacado em modelagem e engenharia de processos de negócio [8]. Analisando os princípios de projeto e modelos de execução no gerenciamento de *workflow* de negócios, um foco em padrões de controle de fluxo de tarefas e eventos torna-se aparente, enquanto o fluxo de dados geralmente fica relegado.

Por outro lado, sistemas de *workflows* científicos tendem a ter modelos de execução orientados a dados, seja em sistemas acadêmicos (por exemplo: KEPLER [22], Taverna [26] e Triana [27]), ou em sistemas comerciais (DiscoveryNet [29], Pipeline-Pilot [30] e LabView [31]). *Workflows* científicos aproximam-se mais de aplicações orientadas a fluxo de dados do que de aplicações de *workflow* de negócios.

A diferença entre sistemas orientados a fluxo de dados e sistemas de controle de fluxo de tarefas pode ser observada em várias representações utilizadas para modelagem desses sistemas. Por exemplo, visualizações de *workflows* de negócios freqüentemente se assemelham a fluxogramas ou diagramas de transição de estados enfatizando eventos e controle de fluxo de tarefas ao invés de dados. A análise formal de *workflows* geralmente envolve o estudo de seus padrões de controle de fluxo de tarefas [2] e freqüentemente é representada por redes de Petri. Já o modelo de execução de sistemas de *workflows* científicos geralmente se assemelha a redes de processamento de fluxo de dados [32] e são representados por grafos com vértices que podem ser vistos como processos concorrentes trocando dados por meio dos arcos. Essas redes de processamento de fluxo de dados possuem, tradicionalmente, aplicações em processamento digital de sinais e engenharia elétrica.

Em [24], os autores definem um modelo formal para *workflows* científicos embasado em modelagem e projeto orientado a atores, originalmente desenvolvido para modelagem de sistemas concorrentes por Hewitt (1973) e, depois, estendido e formalizado por Agha (1985) [33]. Esse modelo formal é proposto como base da

ferramenta KEPLER e será utilizado, neste trabalho, como ponto de partida para compreensão dos fundamentos apresentados pelos autores em [24] para abordagem de *workflows* científicos.

### 2.3.1. Grafos de *workflow* hierárquico orientado a atores

O modelo de atores proposto por Hewitt e Agha é composto por um conjunto de entidades autônomas chamadas atores, componentes ativos com comportamento definido comunicando-se por passagem assíncrona de mensagens. Em grafos de *workflow* orientado a atores, a comunicação entre essas entidades é realizada pela passagem de dados, não necessariamente assíncrona. Bowers e Ludascher apresentam esse modelo em [24] e separam suas definições em dois aspectos distintos: 1) a estrutura de comunicação entre os atores (fluxo de dados), que será definida nesta Seção; 2) a coordenação total do *workflow* (orquestração), definindo a natureza da comunicação entre as entidades, como por exemplo: se o envio de dados é síncrono ou assíncrono. Esse aspecto será aprofundado na Seção 2.3.2.

**Definição 2.3.1** *Grafo de workflow*: é dado por $W = (\mathbf{A}, \mathbf{D})$, com conjunto de vértices **A** composto por atores e conjunto de hiperarestas [2] **D** composto por conexões de fluxo de dados.

**Definição 2.3.2** *Atores*: entidades representando componentes ou tarefas que processam os dados recebidos, ou criadores e emissores de dados a outros atores por uma interface de comunicação chamada conjunto de *portas*.

**Definição 2.3.3** *Portas*: interface de entrada e saída de dados de um ator. Para todo ator $a \in \mathbf{A}$ existe um conjunto associado *portas(a)* de portas de dados, sendo que, para cada $p \in$ *portas(a)* ou *p* é uma porta de entrada ou *p* é uma porta de saída, isto é, *portas(a)* é a união disjunta das portas de entrada e portas de saída de *a*.

**Definição 2.3.4** *Conexões de fluxo de dados*: conectam atores por suas portas de dados para comunicação entre eles por passagem de dados. Seja *entradas(W)* $= \cup_{a \in \mathbf{A}}$ *entradas(a)* o conjunto de todas as portas de entrada do grafo de *workflow* W; os conjuntos *saídas(W)* e *portas(W)* são definidos analogamente. Uma conexão de fluxo de dados $d \in \mathbf{D}$ é a hiperaresta dirigida $d = (\mathbf{O}, \mathbf{I})$, simultaneamente conectando *n* portas de saída $\mathbf{O} = \{o_1, o_2, ..., o_n\} \subseteq$ *saídas(W)* com *m* portas de entrada $\mathbf{I} = \{i_1, ..., i_m\} \subseteq$ *entradas(W)*. Uma conexão de fluxo de dados $d = (\mathbf{O},$

2 *Hiperaresta* de um grafo G = (V, A) é um par ordenado (X, Y) de subconjuntos disjuntos de vértices de V, ou seja, é uma aresta de A que interliga *n* vértices de X a *m* vértices de Y.

**I**) compreende uma etapa de junção, combinando dados das portas de saída **O** e, posteriormente, uma etapa de distribuição que entrega os dados nas portas de entrada **I**[3].

Uma conexão de fluxo de dados $d = (\{o_1\}, \{i_1\})$ entre uma única porta de saída e uma única porta da entrada corresponde a uma aresta dirigida $o_1 \xrightarrow{d} i_1$.Geralmente *d* é representado como nó auxiliar da conexão com *n* arestas chegando de todas as portas de saída $o \in$ **O** e *m* arestas saindo para todas as portas de entrada $i \in$ **I**.

A Figura 2.5 exemplifica um grafo de *workflow* apresentando o conjunto de atores **A**, com suas portas de entrada e saída, e conexões de fluxo de dados **D** entre essas portas.

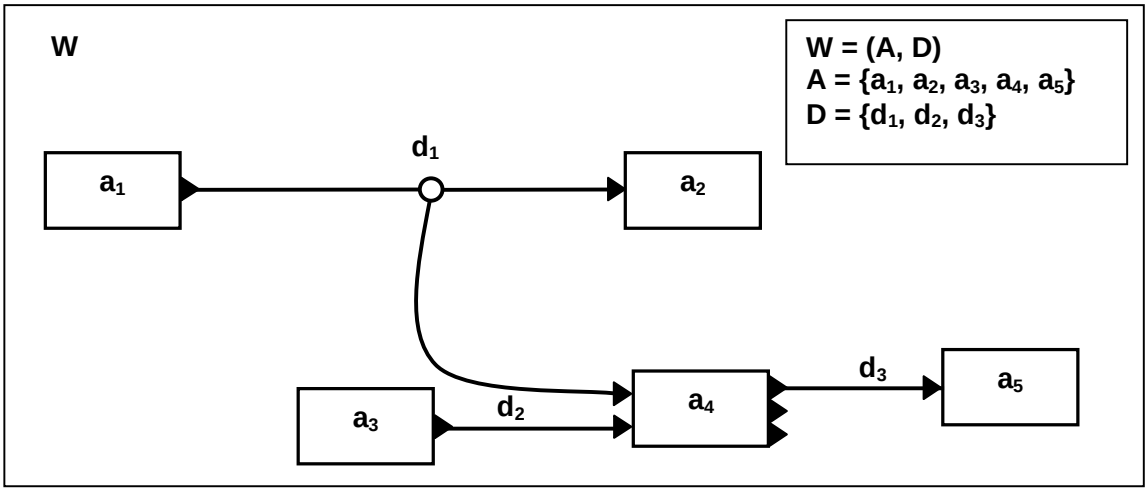



Figura 2.5. Grafo de *workflow*

**Definição 2.3.5** *Abstração e refinamento de workflow*: abstração e refinamento são primitivas cruciais de modelagem. Quando se abstrai um grafo de *workflow* W, quer-se representá-lo em um único ator composto $a_W$. Inversamente, pode-se querer refinar um ator *a* especificando-o mais por meio de um *subworkflow* $W_a$, tornando-o desse modo um ator composto com $W_a$ dentro dele. Em ambos os casos, deve-se certificar de que a assinatura de entrada e saída $\sum_a$ do ator composto combina com a assinatura de entrada e saída $\sum_w$ do *subworkflow* contido. Segue-se exemplo na Figura 2.6.

**Definição 2.3.6** *Portas Livres.* Seja *W* = (**A, D**) um grafo de *workflow*. As portas livres de *W* são todas as portas *p* sem participação em nenhuma conexão de dados *d*, isto é, *portaslivres(W)* := $\{p \mid \forall d \in \mathbf{D}: p \notin d\}$.

[3] A semântica de junção e distribuição de fichas de dados diz respeito à coordenação do *workflow* (orquestração) e será discutida na seção 2.3.2.

**Definição 2.3.7** *Ator composto:* um ator composto $a_W$ é um par (W, $\sum_W$), compreendendo um *subworkflow* de *W* e um conjunto de portas distintas $\sum_W \subseteq$ *portaslivres(W)*, a assinatura de entrada e saída de *W*. Requer-se combinação entre as assinaturas de entrada e saída do *subworkflow* de W e o ator composto $a_W$ contendo *W*, isto é, $\sum_W$ = *portas*($a_W$ ).

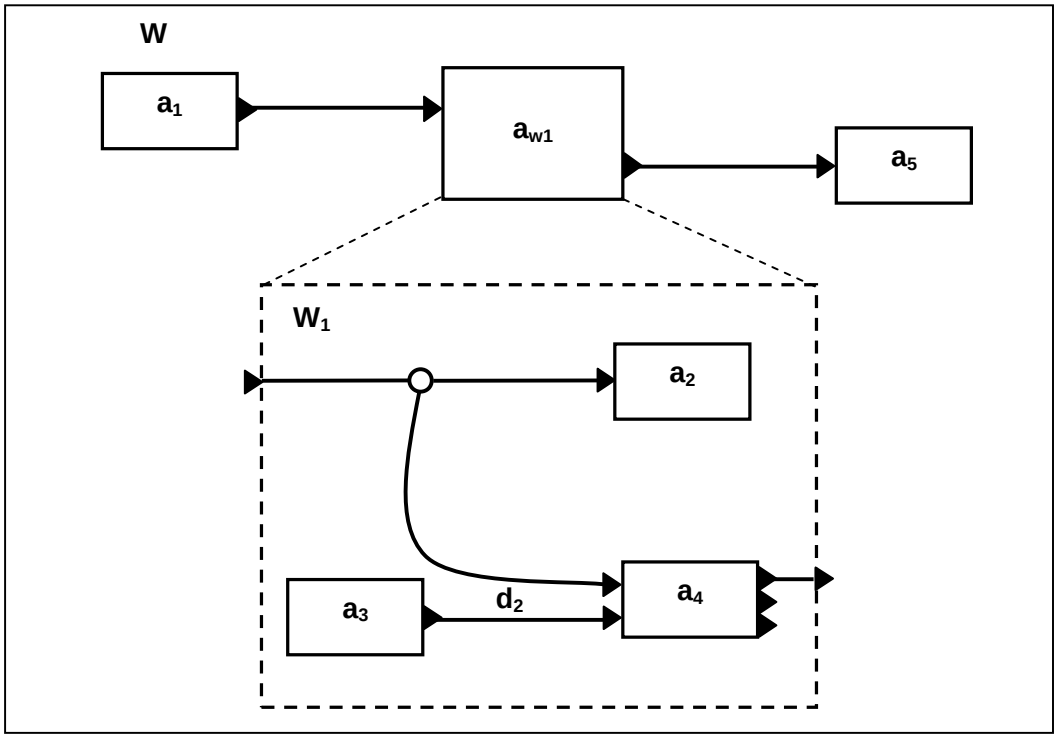



Figura 2.6. Grafo de *workflow* – Abstração ($a_{w1}$) e Refinamento ($W_1$)

**Definição 2.3.8** *Grafos de Workflow hierárquico:* um *workflow* hierárquico *W* = (**A, D**, ∑) é definido como um grafo de *workflow* em que atores podem ser compostos. Indutivamente, *subworkflows* podem ser hierárquicos, portanto, qualquer nível de aninhamento pode ser modelado. Para uniformidade, também se inclui a assinatura de entrada e saída ∑ do nível mais alto do *workflow*.

### 2.3.2. Modelos de computação

Os grafos de *workflow* orientados a ator apresentados anteriormente especificam apenas as ligações de comunicação (fluxo de dados) entre componentes ou tarefas (representados por atores) e, em caso de *workflows* hierárquicos, sua estrutura aninhada por meio do ator composto. Entretanto, o modelo de computação de

*workflows* pode ser escolhido por especificidade e necessidade de cada forma de orquestração das tarefas a serem realizadas. No sistema KEPLER [22], tal coordenação é realizada por uma primitiva de modelagem chamada de *diretor*.

O diretor é a entidade que supervisiona os atores e direciona as ações na execução de um *workflow* (orquestração). Sua principal tarefa é informar aos atores quando converter entrada em saída (execução). As tarefas básicas de um diretor são:

- Conhecer o modo de conexão entre os atores;
- Mover os dados de um ator para outro;
- Decidir e informar aos atores quando devem executar.

**Definição 2.3.9 Grafos de *Workflow* e Modelos de Computação**. A definição de grafos de *workflow W* pode se estendida para incluir o modelo de computação por meio do diretor *M*, isto é, W = (**A, D,** $\sum$, *M*). Em caso de não-especificação da semântica de junção e distribuição de um nó de conexão de dados *d* = (**O, I**), um diretor *M* pode prescrevê-la.

O sistema KEPLER fornece uma variedade de diretores que implementam diferentes semânticas. Os dois diretores de maior interesse neste trabalho, por possuírem o conceito de seqüência de ordem de execução, são: *Synchronous Data Flow* (SDF), com seqüência bem definida de eventos, mas requerendo disponibilidade dos dados nas entradas antes do processamento; e *Process Network* (PN), em que a disponibilidade de dados determina a ordem em que os atores executam. Mais detalhes serão abordados na Seção 3.2.2.

O diretor que trabalha com conceito de tempo, que é um pouco mais complexo que ordem de eventos, chama-se *Continuous Time* (CT). Variáveis descrevem as freqüências (taxas) de ocorrência de eventos (execução dos atores).

O diretor de evento discreto chamado *Discrete Event* (DE), considera a ordem e o tempo em que os eventos ocorrem. Os atores se comunicam através de fichas que representam seqüências de eventos localizados no tempo, ao longo de uma linha em tempo real. Cada ficha é composta de um valor e um rótulo de tempo que especificam a ordem cronológica em que as fichas devem ser processadas por seus respectivos receptores.

## 2.4. Conclusão

Ambos os formalismos apresentados nas seções anteriores deste capítulo são utilizados para representação de *workflow*s, o primeiro em ambientes de negócios e o segundo em ambientes científicos. Cada um possui sua

terminologia própria para representar seus elementos básicos. Porém, como este trabalho visa caracterizar e delimitar o poder de representação de transferência e controle de dados em cada formalismo, requer-se equivalência com terminologia padrão, apresentada na Seção 2.1.

Segue-se tabela resumo da terminologia para cada formalismo, relacionando-a com os conceitos de *workflow*.

| **Componentes de *workflow*** | **Redes de Petri** | **Grafos de *workflow* orientado a atores** |
|---|---|---|
| Tarefas | Transições | Atores |
| -- | Lugares (de entrada e saída) | Portas (de entrada e saída) |
| Canal de dados | Arcos dirigidos | Conexão de fluxo de dados |
| Canal de controle | Arcos dirigidos | Diretores |
| *Subworkflow* | Submodelo | *Subworkflow* |
| Bloco de tarefas | Supernodo | Ator composto |
| Variáveis de Dados | Marcas, Fichas ou *Tokens* | Fichas de dados ou *Tokens* |

Tabela 2.1 Equivalência de terminologias

**Capítulo 3**

# Ferramentas para *workflows*

Neste capítulo, serão apresentadas as ferramentas *CPN Tools* e KEPLER descrevendo as principais características de cada uma e seus elementos de representação dos formalismos apresentados no Capítulo 2.

## 3.1. *CPN Tools*

*CPN Tools* [19] é uma ferramenta para editar, simular e analisar redes de Petri coloridas (rdPc) e pretende substituir *Design/CPN* [18], pacote de *software* difundido para rdPc. *Design/CPN* foi lançado em 1989, para edição e simulação de rdPc. Desde então, um tempo considerável foi investido em desenvolvimento para simulação, geração e análise de espaços de estados completos, parciais e reduzidos.

A ferramenta *CPN Tools* é o resultado do projeto de pesquisa CPN2000 [20], da universidade de Aarhus, patrocinada pelo Danish National Centre for IT Research (CIT), George Mason University, Hewlett-Packard, Nokia e Microsoft. Essa ferramenta possibilita o projeto de rdPc, ou seja, modelos com tipos de dados complexos (conjunto de cores) e manipulações complexas de dados (expressões de guarda e de arco). A próxima seção apresenta os elementos básicos de construção e edição de rdPc usando *CPN Tools*.

### 3.1.1. Construção e edição de rdPc

O gerenciamento de ambiente de trabalho do *CPN Tools* (Figura 3.1) facilita controlar o grande número de páginas tipicamente encontradas em rdPc industriais. O *espaço de trabalho* ocupa a tela inteira e contém janelas chamadas pastas (*binders*) com folhas. As pastas reduzem o número de janelas na tela e o tempo gasto com organização. É possível organizar o trabalho agrupando as folhas relacionadas, reduzindo o tempo gasto procurando janelas escondidas.

O *índice* à esquerda do espaço de trabalho lista todas as ferramentas disponíveis e os elementos de rede no *CPN Tools* (Figura 3.1). Assemelha-se a árvore de arquivos do *Windows Explorer*, e o usuário pode arrastar palhetas de ferramentas, páginas de rede de Petri colorida (*New Page*), ou declarações de conjuntos de cores (*colset*) ou variáveis (*var*) para o espaço de trabalho. No exemplo abaixo, as palhetas *Create* e *Sim* foram arrastadas do índice para o espaço de trabalho.

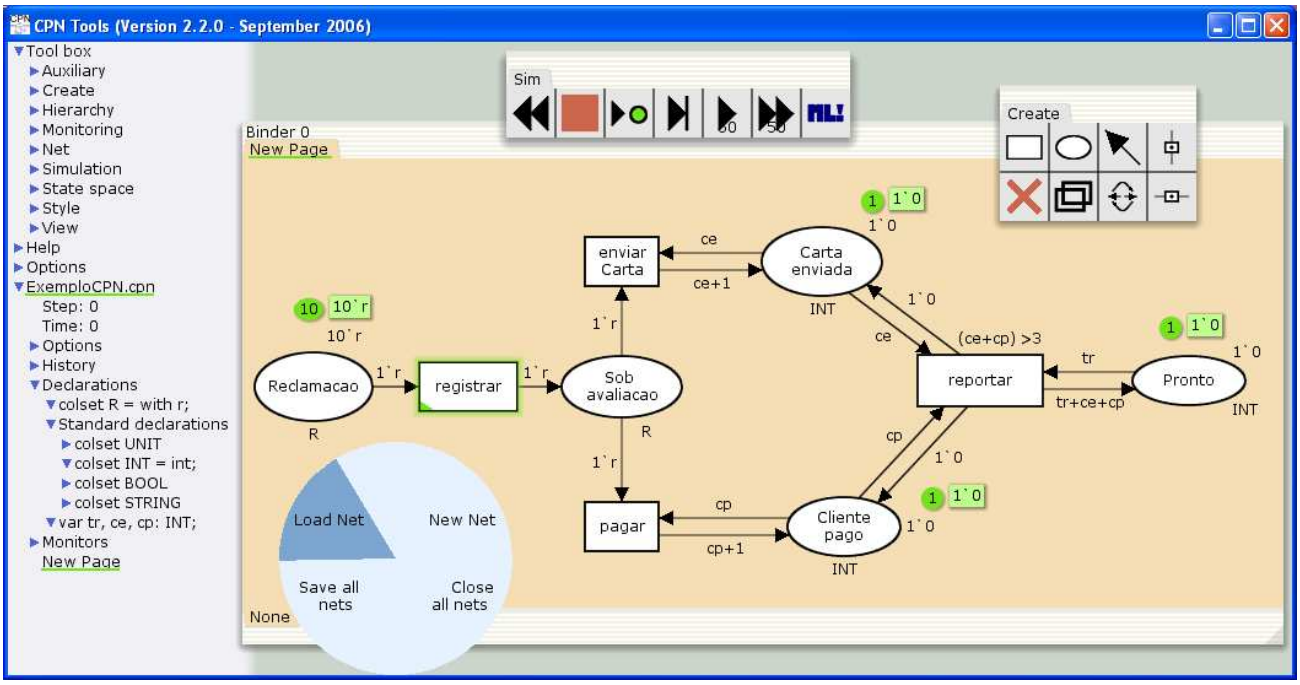



Figura 3.1. Interface *CPN Tools*

Com a ferramenta *Create* criam-se elementos rdPc, isto é, lugares (elipses), transições (retângulos) e arcos (setas). Todos os elementos da rede podem ser criados usando palhetas ou menus de marcação (clique com botão direito). Esses elementos podem ser posicionados livremente dentro de uma folha, ou ligados às linhas guias magnéticas mantendo os objetos alinhados.

A *adição de inscrições* (nomes, expressões e tipos de cores associados aos lugares) aos elementos da rede é feita clicando sobre o elemento. Isso selecionará a inscrição-padrão do elemento, por exemplo, o nome de um lugar ou a expressão de um arco. Para editar as demais inscrições de um mesmo objeto, a tecla TAB pode ser usada para se mover de uma inscrição para outra do objeto em questão. A verificação de sintaxe do modelo em *CPN Tools* é feita automaticamente a cada declaração ou inscrição editada.

### 3.1.2. Processos de negócios em *CPN Tools*

A Figura 3.2 apresenta o modelo do "processo de atendimento a reclamações", apresentado na Seção 2.2.1, implementado pela ferramenta *CPN Tools* em sua marcação inicial. Apenas a transição *registrar* está inicialmente habilitada para disparo, pois as fichas (10`r) em seu lugar de entrada (*Reclamação*) satisfazem a expressão de arco (1`r).

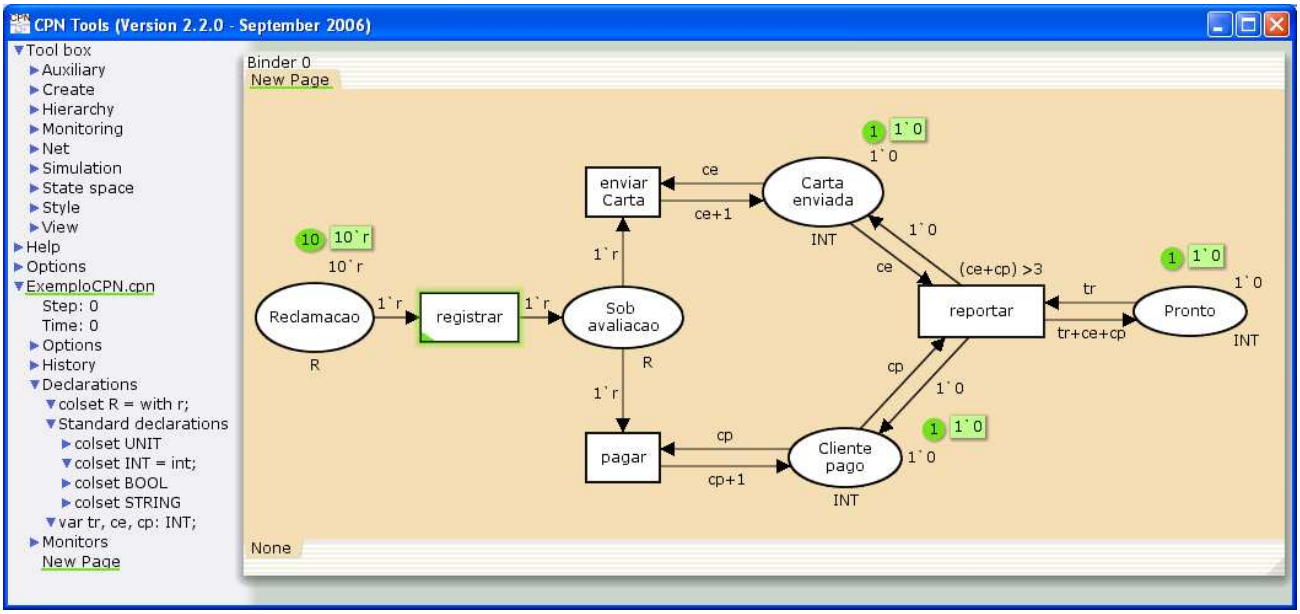



Figura 3.2. Exemplo de atendimento de reclamações usando *CPN Tools*

As simulações são controladas pela ferramenta de simulação: palheta *Sim*. Os ícones para as ferramentas da simulação assemelham-se a teclas de um gravador de vídeo cassete (ver Figura 3.1). As teclas possuem as seguintes funções ordenadas da esquerda para direita:

1ª) Mais à esquerda: retorna uma rdPc a sua marcação inicial;

2ª) Pára uma simulação de rdPc;

3ª) Permite a escolha manual de uma transição a ser disparada e o valor associado às variáveis em questão;

4ª) Dispara uma transição habilitada;

5ª) Realiza um determinado número de transições exibindo as marcações intermediárias;

6ª) Realiza determinada quantidade de transições sem exibir as marcações intermediárias.

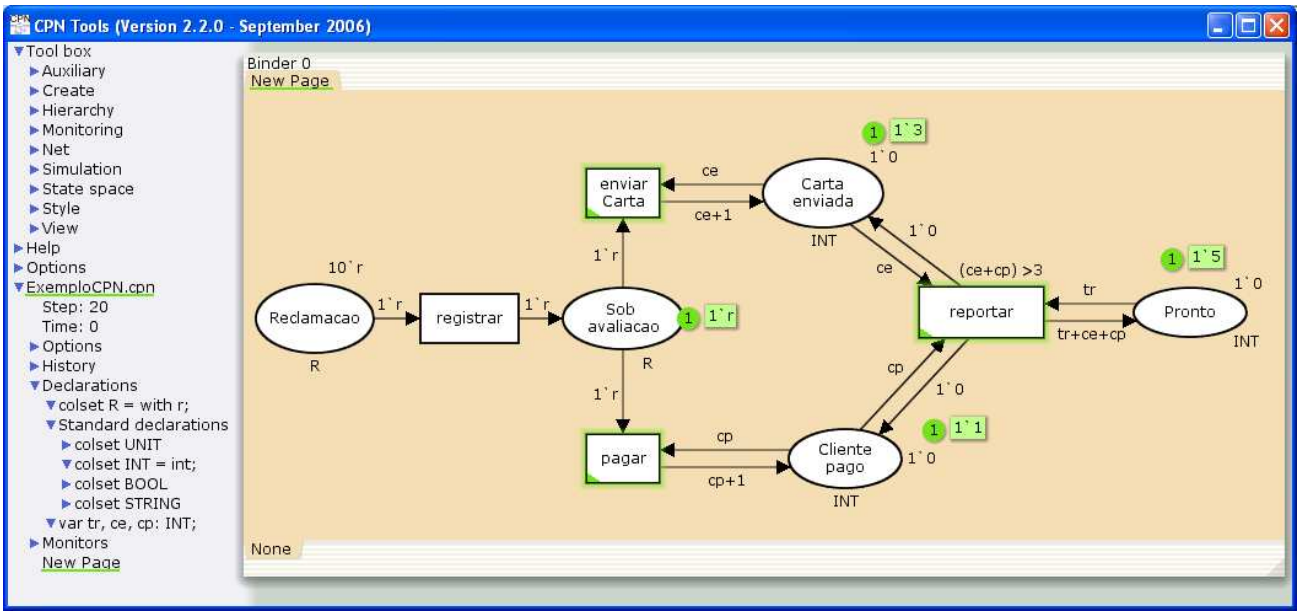



Figura 3.3. Exemplo de atendimento de reclamações após 20 transições

Na Figura 3.3 visualiza-se o estado da rede após 20 disparos de transição simulados pela ferramenta *CPN Tools*, habilitando a transição *reportar*, pois o valor da ficha em *Carta enviada* (ce = 3) mais o valor da ficha em *Cliente pago* (cp = 1) ultrapassa três, atendendo a expressão de guarda (ce+cp)>3.

## 3.2. Sistema KEPLER

Muitos cientistas conduzem análises em ambientes diferentes de *hardware* e *software*, coordenando manualmente a exportação e importação de dados de um ambiente para outro. O sistema de *workflow* científico KEPLER, agilizando a criação e execução de *workflows*, é de uso prático aos cientistas, que assim podem projetar, executar, monitorar, re-executar e comunicar procedimentos analíticos repetidamente com esforço mínimo. Esse sistema combina projeto de *workflow* em alto nível com execução e interação em tempo de execução, acesso a dados remotos e locais e serviços de invocação remota e local.

KEPLER foi construído a partir do sistema orientado a fluxo de dados PTOLEMY II [28], herdando suas características e acrescentando outras novas para apoiar *workflows* científicos. Ambos estão fundamentados em modelo formal de *workflow* científico embasado em modelagem e projeto orientado a atores, originalmente desenvolvido para modelagem de sistemas concorrentes complexos.

### 3.2.1. Construção e edição de *workflow*s científicos

Nesta seção, será apresentada uma coleção de primitivas básicas de desenvolvimento de *workflow* científico orientado a atores. Cada primitiva corresponde a uma operação básica do modelo formal. Primitivas podem ser descritas como transformações resultantes da aplicação de operações ao *workflow*.

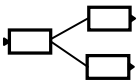
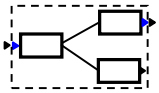
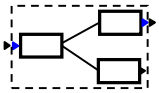
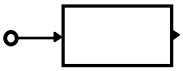
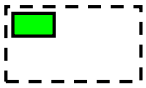
| *Transformação Básica* | *Workflow inicial* | *Workflow resultante 1* | *Workflow Resultante 2* |
|---|---|---|---|
| *t1*: Introdução de entidade (ator ou conexão) | |  |  |
| *t2*: Introdução de porta |  |  |  |
| *t3*: Abstração hierárquica |  |  | |
| *t4*: Refinamento hierárquico |  |  | |
| *t5*: Conexão de fluxo de dados |  |  |  |
| *t6*: Introdução de diretor |  |  | |

Tabela 3.1 Primitivas básicas de projeto orientado a ator [24]

A Tabela 3.1 resume as primitivas: apresenta a transformação associada a cada primitiva (primeira coluna) e os possíveis resultados (colunas 3 e 4) de sua aplicação num dado *workflow* inicial (coluna 2). São elas:

*t1:* Inserir novo ator e novas conexões de fluxo de dados ao *workflow*;

*t2:* Adicionar portas de entrada e saída aos atores;

*t3:* Agrupar uma parte do *workflow* (abstração) em um ator composto;

*t4:* Definir um ator enquanto composição;

*t5:* Criação de conexão de fluxo de dados;

*t6:* Associar um diretor a um *workflow*.

Atores do modelo proposto por Agha [33] só podem enviar mensagens para atores conhecidos - cujos endereços lhes foram dados em tempo de criação, ou recebidos por mensagem, ou atores que ele próprio criou. No sistema KEPLER, a restrição equivalente é que um ator só pode enviar uma mensagem para outro se existir referência à porta de entrada do último. Essa referência é obtida pela topologia do *workflow*, ou seja, verificando-se as portas conectadas.

O sistema KEPLER possui uma interface gráfica e bibliotecas de componentes para criação de workflows científicos executáveis. Para desenvolver um sistema de workflow científico no Kepler, o usuário utiliza a interface gráfica para arrastar e colocar os componentes na área de desenvolvimento, conectando-os de maneira a obter o fluxo de dados desejado.

Novos atores podem ser codificados e adicionados a biblioteca local do sistema. Atores compostos podem ser definidos e implementados a partir de outros atores já disponíveis nas bibliotecas. Segue exemplo de projeto e implementação de *workflow* científico.

### 3.2.2. *Workflow* científico no sistema KEPLER

A Figura 3.4 apresenta um exemplo simples de *workflow* científico que utiliza componentes disponíveis na biblioteca KEPLER como atores. Apenas o ator composto *Estatística* foi criado por usuário. Essa implementação lê uma lista de números inteiros do arquivo especificado em *File Reader*, calcula parâmetros estatísticos básicos (média, desvio-padrão e variância) e grava os resultados em arquivo, especificado em *File Writer*. O ator composto *Estatística* contém um *subworkflow* que calcula os valores estatísticos, convertendo-os em uma lista de valores (vetor). Os atores *Expression To Token* e *Token To Expression* foram utilizados para converter os valores do tipo *String* para valores numéricos e vice-versa. As portas de saída dos atores *Expression To Token* e *Estatística* estão conectadas a atores do tipo *Display*, chamados *Array Entrada* e *Array Saída*, que exibem na tela a lista de valores de entrada e saída, respectivamente.

Cada ação é representada pelos atores e estes se comunicam por meio de canais (conexões de fluxo de dados). A execução total do *workflow* é orquestrada por um diretor de fluxo de dados síncrono (SDF).

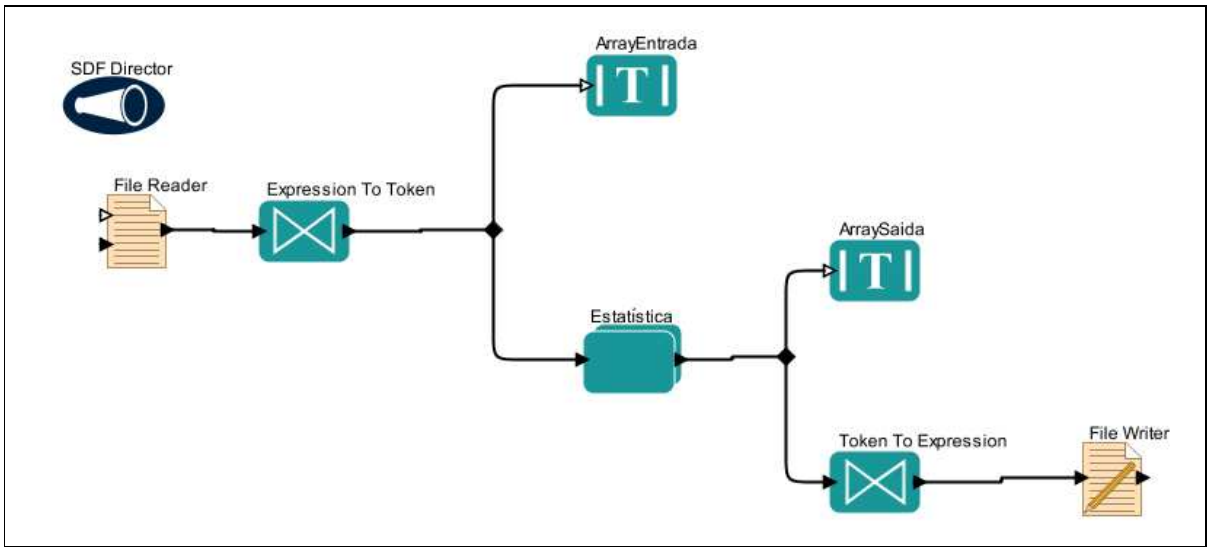



Figura 3.4. Exemplo de *workflow* orientado a ator representado no sistema KEPLER

O mecanismo de ocultar detalhes de um *subworkflow* em um componente chamado ator composto é essencial para moderar a complexidade. Esse conceito é representado pelo ator *Estatística* detalhado na Figura 3.5.

O ator *SummaryStatistics* recebe uma lista de inteiros em sua porta de entrada, que se comunica com a porta de entrada do *subworkflow*. Este ator calcula e envia para sua saída os parâmetros: média, desvio-padrão e variância dos valores de entrada. Esses resultados são recebidos na porta de entrada do ator *Elements to Array* e transformados em uma lista de valores (vetor), enviada para sua porta de saída, que se conecta à porta de saída do *subworkflow* (port2).

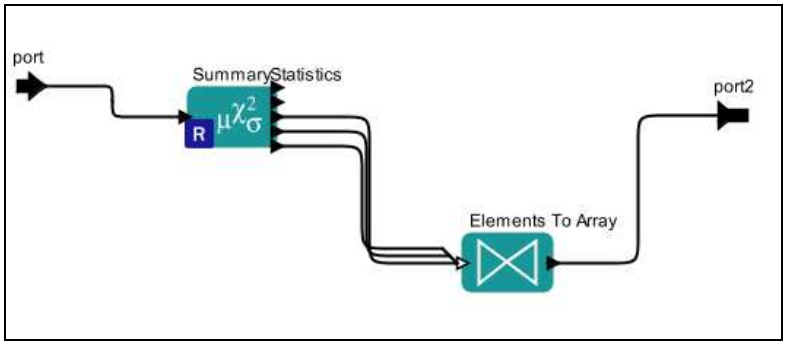



Figura 3.5. Detalhamento do ator composto *Estatística*

O modelo de computação, definido por meio da escolha de um diretor para o *workflow*, especifica todos os comportamentos de comunicação entre atores, separando a preocupação de orquestrar (diretor) da preocupação de executar (ator).

Com o diretor *Process Network* - PN, o sistema é modelado como uma rede de processos, ou seja, o diretor executa cada ator como um processo (linha de execução) independente, mas que se comunica com outros por passagem de mensagens (seqüências de fichas de dados). Para isso, são utilizadas as conexões de dados – com canais unidirecionais do tipo FIFO (*first-in-first-out*). Tal comunicação é assíncrona, ou seja, o emissor não precisa esperar que o receptor esteja pronto para enviar fichas de dados. Daí a importância do uso de canais do tipo FIFO para armazenar as fichas de dados que chegam às portas de entrada dos atores.

Nesse modelo, os atores representam funções mapeando seqüências de fichas de entrada em seqüências de saída. Uma tarefa fica bloqueada tentando ler seu canal de dados vazio, até que a mensagem esteja disponível, ou seja, os dados apresentem-se em suas portas de entrada.

Esse diretor é mais indicado para modelagem de sistemas paralelos ou que necessitam de controle de execução dinâmico, ou seja, com ocorrência de tarefas determinada em tempo de execução. Vários exemplos apresentados na Seção 4.2 utilizam diretor PN.

Usando-se o diretor *Syncronous Dataflow* - SDF, uma especialização de *Process Network*, atores enviam ou recebem uma quantidade fixa de fichas de dados a cada disparo (execução). A ordem de disparo dos atores é determinada estatisticamente antes da execução do modelo, gerando um plano de execução (*schedule*) antes da primeira iteração do *workflow*. Essa ordenação garante que um ator só realiza sua tarefa após a execução de atores cujos dados de saída são necessários em sua entrada.

O uso diretor SDF resulta numa execução com sobrecarga mínima, assim como menos desperdício de memória e impossibilita a ocorrência de um impasse. Apesar da execução mais eficiente, o SDF é estático, ou seja, o mesmo número de fichas é consumido a cada iteração por todos os atores. Isso impede o uso de estrutura de controle dinâmica, pois ela pode decidir, em tempo de execução, quais atores devem executar e a quantidade de fichas que será consumida. No exemplo da Figura 3.4, apresenta-se um modelo simples, no qual a seqüência de tarefas é facilmente determinada pelo diretor SDF. Outros exemplos são apresentados na Seção 4.2.

## 3.3. Conclusão

Neste capítulo, foram apresentadas as ferramentas *CPN Tools* e KEPLER, que se propõem a implementar os formalismos apresentados no Capítulo 2. Elas serão comparadas no próximo capítulo, com base nos padrões de dados propostos por Russel e Aalst.

**Capítulo 4**

# Padrões de controle de dados em *CPN Tools* e KEPLER

Neste capítulo, são comparadas as ferramentas *CPN Tools* e KEPLER. Os padrões de controle de dados de referência são abordados na primeira seção deste Capítulo.

Na segunda seção, são apresentadas duas representações para cada um dos padrões comparados. A primeira corresponde a uma possibilidade de representação do padrão utilizando *CPN Tools,* e a segunda, usando KEPLER. Para cada padrão, serão discutidas as características de cada ferramenta que permitem ou impedem a sua representação, abordando, assim, os limites representativos de cada ferramenta e de seu respectivo formalismo.

Para cada padrão, as ferramentas serão classificadas segundo sua capacidade de representá-lo. A ferramenta representará o padrão quando estiverem presentes estruturas e conceitos necessários para isso. A ferramenta não representará o padrão quando faltarem estruturas ou conceitos necessários para a sua representação. Diz-se que a ferramenta representa parcialmente o padrão quando as estruturas e conceitos existentes são suficientes para representar o padrão indiretamente ou apenas parte das situações compreendidas por ele.

Após as comparações, a terceira seção apresenta sinteticamente os resultados obtidos e analisados em tabelas que facilitam a visualização dos resultados.

## 4.1. Padrões de controle de dados

Ao propor uma caracterização e delimitação de representação e controle de dados em ambientes distintos, de negócios e científicos, comparando seus respectivos formalismos, surge o questionamento: como comparar abordagens com terminologias tão diferentes? Porém, apesar das diferenças, processos de negócio e *workflows* científicos possuem suas bases nos conceitos de *workflow*. Assim, a iniciativa de trabalho e pesquisa sobre

padrões de dados de *workflow*, realizado por Russel e Aalst, fornece a base conceitual para análise aprofundada da perspectiva de controle de dados em ambos os formalismos.

Considerando uma série de comportamentos-padrão recorrente em diferentes paradigmas de modelagem de *workflow*, Russel e Aaslt em [1] descrevem 39 padrões de dados que podem ser classificados em quatro grupos:

I) Visibilidade de dados: relacionados com a amplitude e a maneira pela qual os dados podem ser vistos por vários componentes de um *workflow*. Identifica o escopo dos dados em um modelo de *workflow;*

II) Interação de dados: centrados na maneira pela qual os dados são comunicados entre elementos ativos (componentes) dentro de um processo de *workflow* e como esses componentes podem influenciar o tráfego desses dados;

III) Transferência de dados: considera os meios pelos quais a transferência real de dados ocorre entre os componentes do *workflow*. Descreve os diversos mecanismos de transmissão de dados por meio da interface de um componente de *workflow*, caracterizando a passagem de dados por valor ou referência;

IV) Roteamento baseado em dados (controle de fluxo de tarefas): caracteriza a maneira pela qual os dados podem influenciar o funcionamento de outros aspectos do *workflow*, em particular, a perspectiva de controle de fluxo de tarefas.

Para o presente trabalho, foram escolhidos 10 padrões para estudo e comparação de controle de dados nas ferramentas de *workflows* apresentadas no Capítulo 3. Como o enfoque deste trabalho está na caracterização de como cada formalismo e suas respectivas ferramentas representam o envio de dados entre as tarefas do *workflow* e como eles podem influenciar o controle de fluxo das tarefas, os padrões descritos nos grupos I e III imediatamente são descartados por fugirem desse contexto.

Dentre os seis padrões de interação de dados (grupo II) foram selecionados os três padrões mais simples, ficando os demais para aprofundamento em trabalhos futuros. Os sete padrões do quarto grupo - roteamento baseado em dados – serão considerados nessa comparação por relacionarem-se diretamente à influência dos dados no controle de fluxo de tarefas.

Para cada padrão, serão apresentadas as seguintes informações: definição, contendo a descrição do padrão; motivação, descrevendo possíveis situações ou exemplos justificando o uso do padrão; implementação, apresentando possíveis soluções práticas já observadas em distintas ferramentas disponíveis no mercado. Na seqüência, serão apresentados exemplos de implementação do padrão em questão nas ferramentas *CPN Tools* e KEPLER, seguidos da análise da solução dada.

## 4.2. Representação dos padrões de controle de dados

### 4.2.1. Padrões de interação de dados – grupo II

**Padrão 8: Tarefa a tarefa**

**Definição.** Capacidade de comunicar dados entre duas instâncias de tarefas em um mesmo caso.

**Motivação.** A passagem dos elementos de informação (dados) entre tarefas é fundamental em sistemas de *workflow*. Em muitas situações, tarefas realizam suas atividades em seu próprio espaço de memória sem compartilhar dados em uma base global. O não compartilhamento requer transporte de dados entre tarefas distintas, quando estas utilizam dados em comum.

**Implementação.** Todos os sistemas de *workflow* permitem a passagem de parâmetros entre tarefas. No entanto, isso pode ocorrer de várias maneiras diferentes, dependendo da relação entre as perspectivas de fluxo de dados e de controle de fluxo de tarefas no *workflow*. Seguem-se três abordagens para passagem de dados.

1) Canais de controle de fluxo e controle de dados integrados (Figura 4.1): onde ambos, controle de fluxo de tarefas e dados, são transmitidos simultaneamente entre tarefas utilizando o mesmo canal.

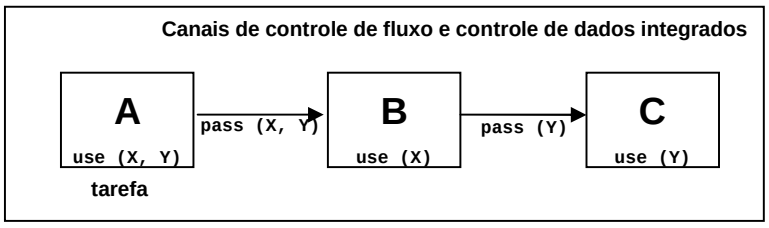



Figura 4.1. Canais de controle de fluxo e controle de dados integrados[1]

2) Canais de dados distintos (Figura 4.2): os dados são transferidos entre tarefas via canais de dados explícitos, que são distintos dos canais de controle de fluxo de tarefas dentro do *workflow*.

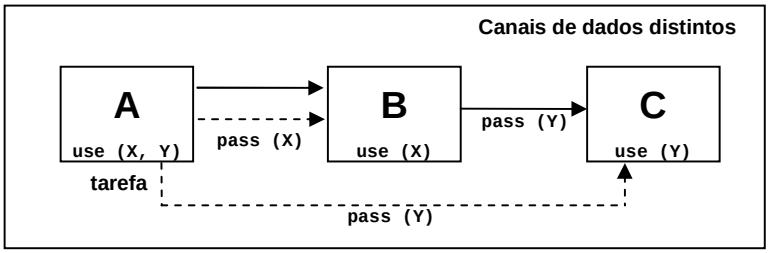



Figura 4.2. Canais de dados distintos [1]

3) Não há passagem de dados (Figura 4.3): tarefas partilham os mesmos dados (por ex.: por meio de acesso a dados compartilhados globalmente), prescindindo de passagem explícita de dados.

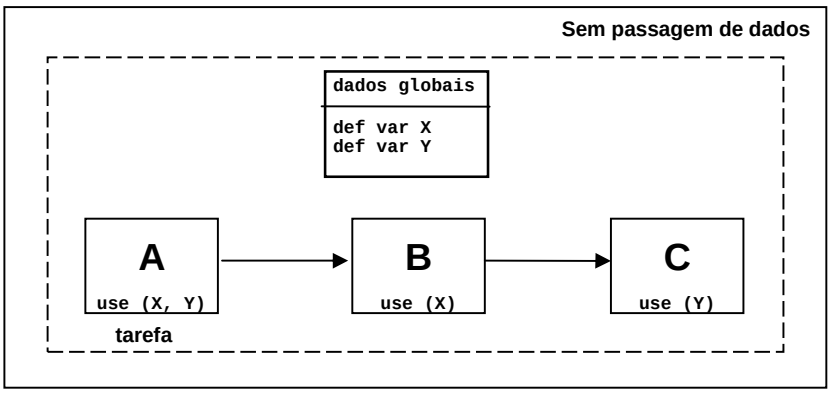



Figura 4.3. Sem passagem de dados [1]

***CPN Tools*:** utiliza a primeira abordagem. Os dados são transmitidos entre tarefas (transições) por meio de arcos e suas expressões. Na Figura 4.4, a expressão *Eng* representa o dado (ficha) a ser consumido pela tarefa (*enviaEngradado* ou *recebeEngradado*) ou produzido em sua saída.

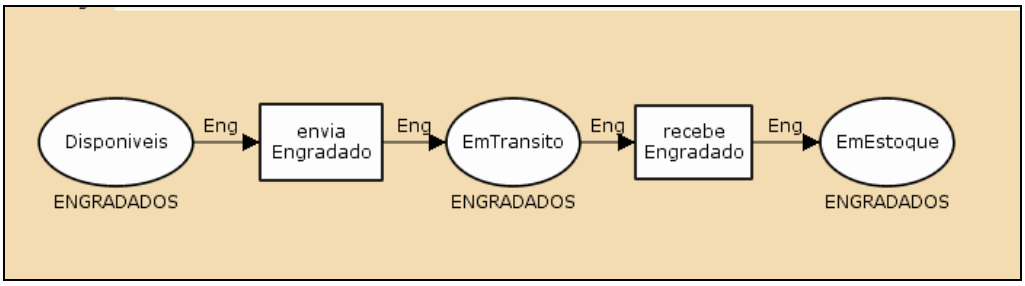



Figura 4.4. Abordagens para interação de dados entre tarefas no *CPN Tools*

**KEPLER:** utiliza a primeira abordagem, por meio das conexões de fluxo de dados (conexão entre atores) e a terceira abordagem quando utiliza parâmetros globais. A Figura 4.5 mostra os dois casos. Os dados dos atores *Ator1*, *Ator2*, *Ator3* para *AddSub* são enviados por meio de suas respectivas conexões. Entre o ator *AddSub* e os atores *Display* e *AtualizaParametro* a mesma implementação se repete. Um parâmetro global, inicialmente valendo 10, é declarado e utilizado pelo *Ator3*. Esse mesmo parâmetro é atualizado em *AtualizaParametro* com o resultado da soma enviado por *AddSub*. Numa segunda execução desse mesmo *workflow*, o parâmetro usado pelo *Ator3* já possui novo valor. Essa implementação demonstra a possibilidade da ferramenta de transmitir valores por meio de parâmetros globais, sem a passagem explícita de dados entre as tarefas.

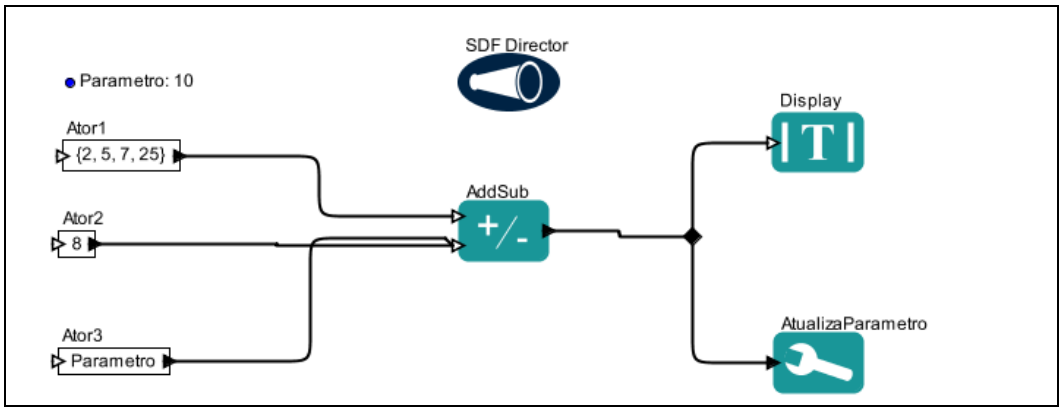



Figura 4.5. Abordagens para interação de dados entre tarefas no sistema KEPLER

**Padrão 9: Bloco de tarefas para *subworkflow***

**Definição.** Capacidade de transmitir dados de uma instância de bloco tarefa (ator composto ou supernodo) ao *subworkflow* correspondente.

**Motivação.** A maioria dos sistemas de *workflow* possui a noção de composição ou blocos de tarefas de alguma forma. São análogos a chamadas de procedimentos em linguagem de programação. Indicam uma tarefa cuja implementação é descrita mais detalhadamente em outro nível de abstração (normalmente em outras partes do projeto de *workflow*). No nível mais alto de abstração, a composição se denomina *bloco de tarefas*, ou *supernodo*, em redes de Petri, ou *ator composto* em KEPLER. Já o detalhamento, é chamado *subworkflow*. A questão que surge quando os dados são passados para um bloco de tarefas é se eles são imediatamente acessíveis por todas as tarefas que compõe a implementação de seu *subworkflow*, ou se os dados devem passar explicitamente entre o bloco de tarefas e os *subworkflows*.

**Implementação.** Normalmente, três abordagens podem ser usadas para tratar a comunicação de parâmetros de um bloco tarefa para sua implementação (*subworkflow*). As características de cada abordagem são as seguintes:

1) Passagem de dados implícita (Figura 4.6): o dado passado ao bloco de tarefas fica imediatamente acessível a todas as sub-tarefas. Dispensa passagem explícita de dados entre o bloco de atividades e sua implementação. Os dados estão acessíveis ao *subworkflow*;

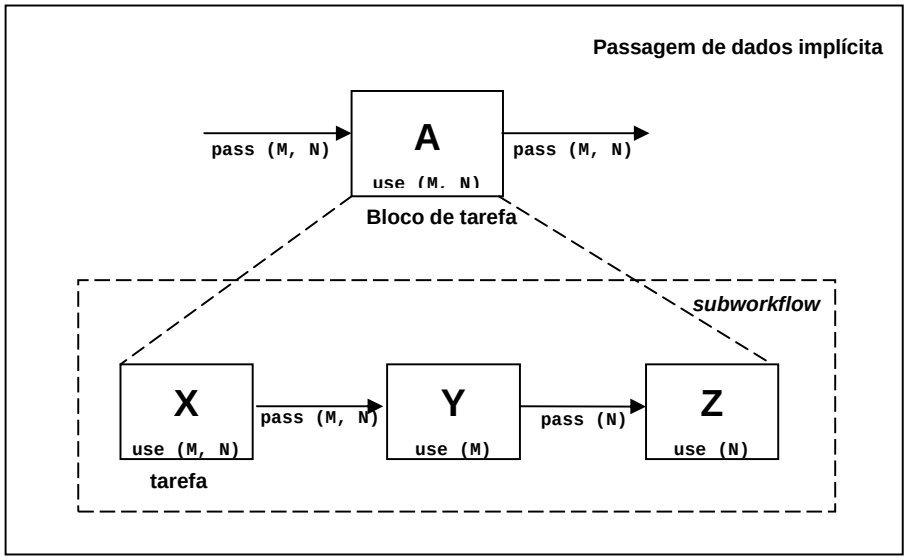



Figura 4.6. Interação bloco de tarefas - *subworkflow* com passagem de dados implícita [1]

2) Passagem de dados explícita via parâmetros (Figura 4.7): dados fornecidos ao bloco de tarefas devem ser especificamente passados como parâmetros para sua implementação (*subworkflow*). Neste caso, os dados do bloco de tarefas são mapeados para dados do *subworkflow*, com nomes distintos. Esta capacidade proporciona certa independência nos nomes de dados do bloco de tarefas e sua implementação (*subworkflow*). Esta situação é particularmente importante quando blocos de tarefas compartilham o mesmo *subworkflow*;

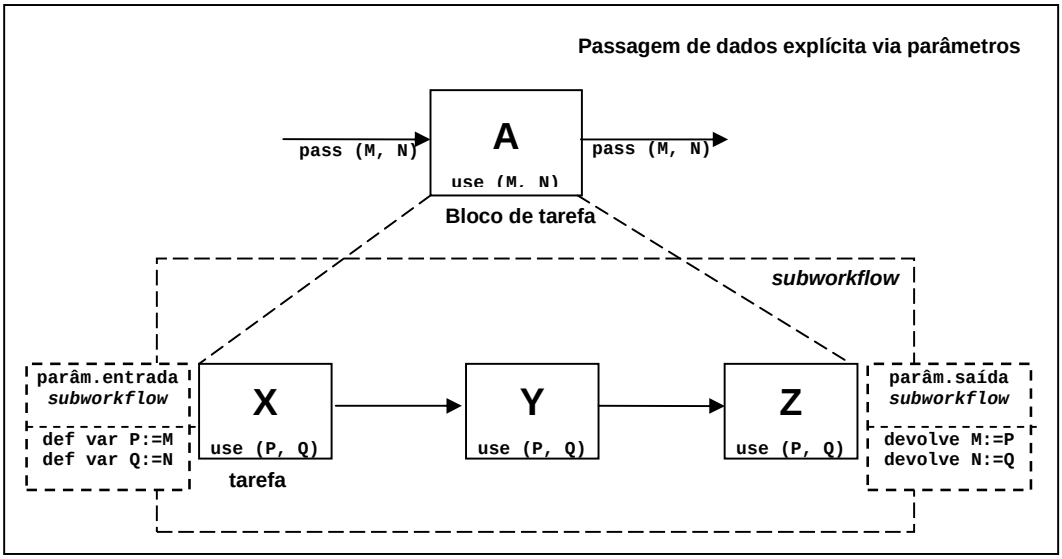



Figura 4.7. Passagem de dados explícita via parâmetros [1]

3) Passagem de dados explícita via canais de dados (Figura 4.8): os dados são passados do bloco de tarefas, especificamente via canais de dados, para todas as tarefas no *subworkflow* que exigem acesso a eles.

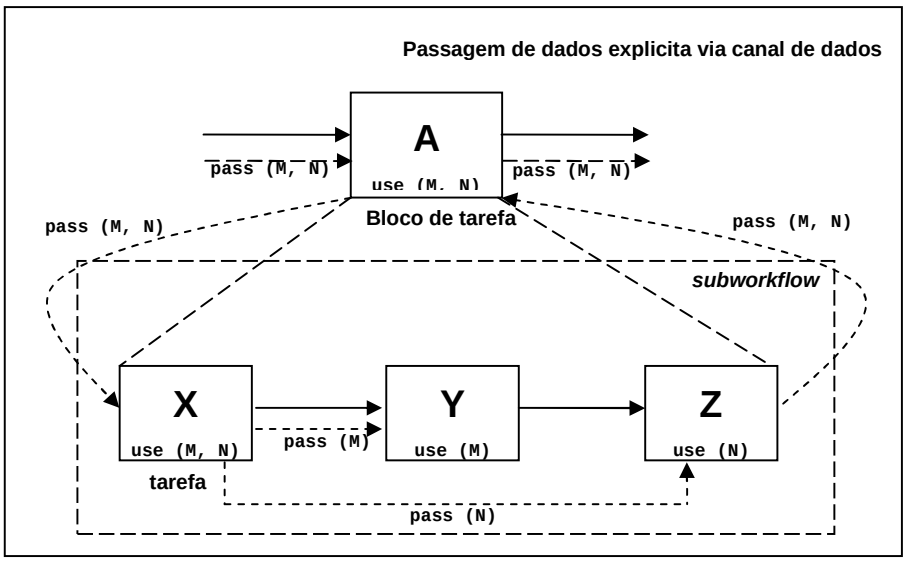



Figura 4.8. Passagem de dados explícita via canais de dados [1]

***CPN Tools***: utiliza a primeira estratégia para passagem de dados para *subworkflow*. O exemplo da Figura 4.9 representa quatro supernodos (*Sender*; *Network*; *RecNo1*; *RecNo2*) que podem ser detalhados em *subworkflows*. Os dados do *workflow* são diretamente acessíveis pelos *subworkflows*.

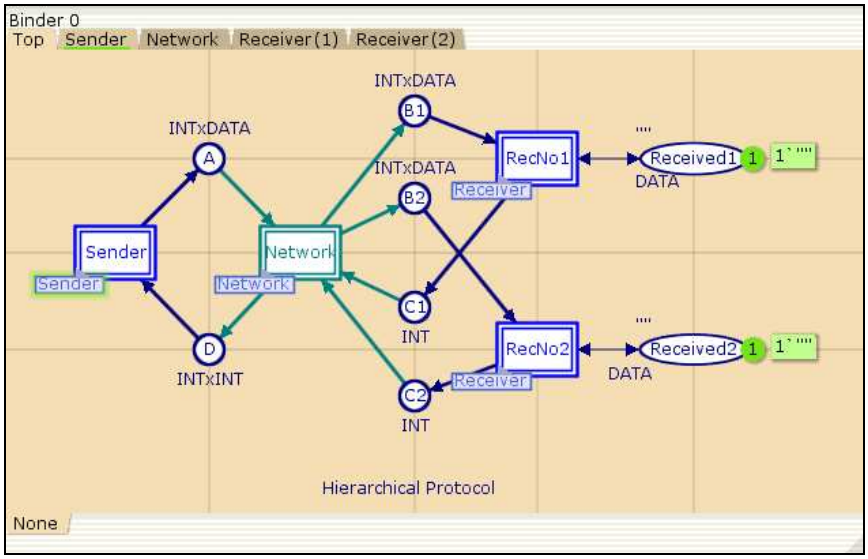



Figura 4.9. Exemplo de blocos de tarefa, ou super-nodos, em *CPN Tools*

A Figura 4.10 detalha o bloco de tarefas *Sender* com acesso aos dados do *workflow* por meio dos lugares *A (out)* e *D(in).*

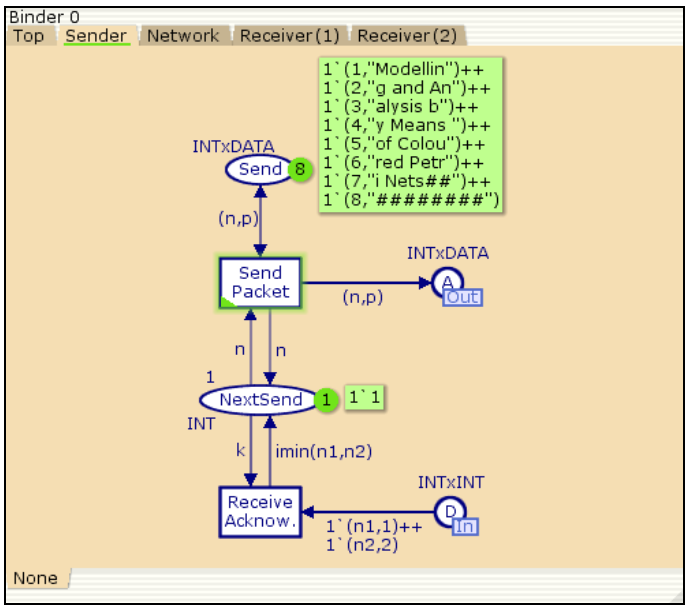



Figura 4.10. *Subworkflow* em *CPN Tools*

**KEPLER**: também utiliza a passagem de dados implícita. No exemplo da Figura 4.11, o bloco de tarefas (ou ator composto) *DemandaEstoque* recebe dados por meio de sua porta de entrada deixando-os automaticamente disponíveis para sua implementação (*subworkflow*).

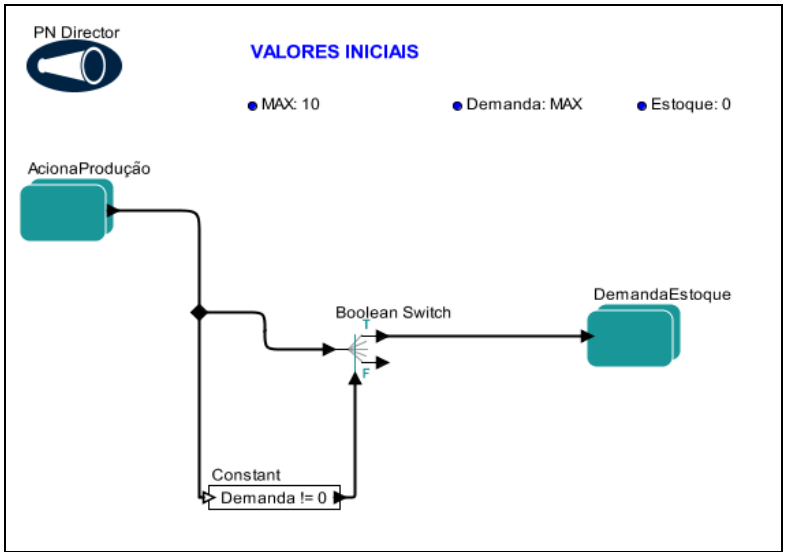



Figura 4.11. Exemplo de blocos de tarefa, ou ator composto no sistema KEPLER

A Figura 4.12 detalha o ator composto *DemandaEstoque*, que por sua vez possui dois outros blocos de tarefas em sua implementação: *AtendeDemanda* e *EstocaEngradado*.

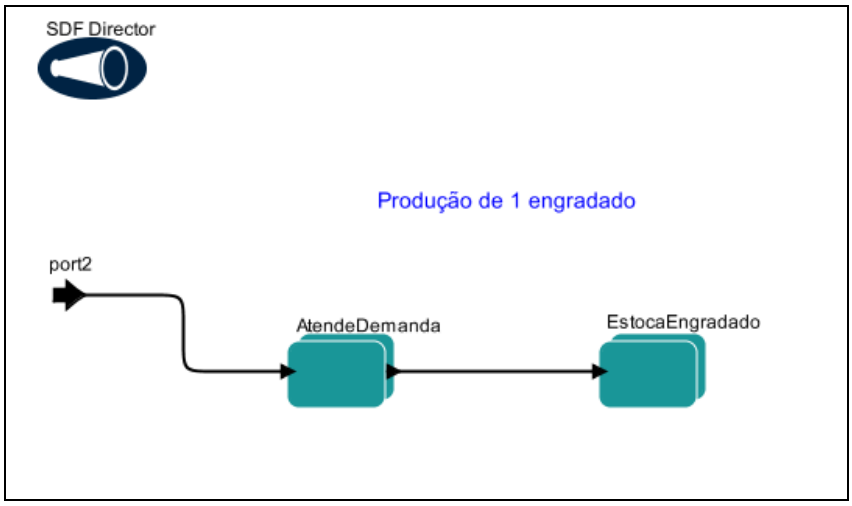



Figura 4.12. *Subworkflow* DemandaEstoque.

Vê-se na Figura 4.12 o bloco de tarefas *AtendeDemanda* receber dados pela porta de entrada (*port*) e disponibilizá-los para sua implementação (Figura 4.13). Após atualização da variável *Demanda*, o dado é enviado para a porta de saída (*port2*), ficando disponível para o bloco de tarefas *EstocaEngradado* (Figura 4.12).

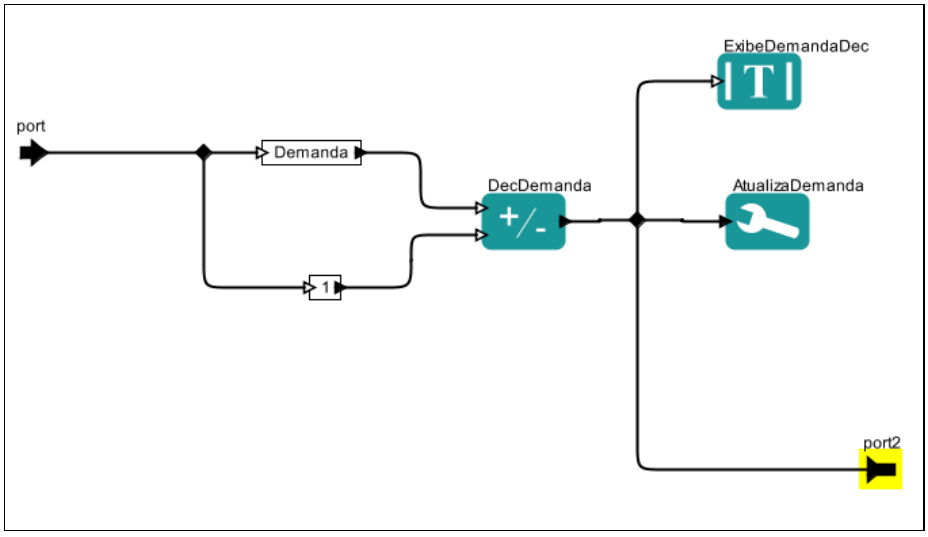



Figura 4.13. *Subworkflow* AtendeDemanda.

**Padrão 10: *Subworkflow* para bloco de tarefas**

**Definição.** Capacidade de transmitir dados a partir de um *subworkflow* de volta para seu bloco de tarefa (instância) correspondente.

**Motivação.** Encerrando a execução do *subworkflow*, os dados ficarão disponíveis para o bloco de atividades que o chamou.

**Implementação.** As abordagens adotadas para lidar com esse padrão são essencialmente as mesmas identificadas pelo Padrão 9. Quando a transmissão dos dados é explícita, um mapeamento deve ser especificado para cada parâmetro de saída indicando quais elementos no nível de bloco de tarefas receberão os respectivos valores de saída.

### 4.2.2. Padrões de roteamento baseado em dados (controle de fluxo de tarefas) - grupo IV

**Padrão 33: Pré-condição de tarefa – existência de dados (Figura 4.14)**

**Definição.** Pré-condições baseadas em dados podem ser especificadas para tarefas dependentes da presença de dados em sua entrada no momento da execução.

**Motivação.** É desejável a habilidade para lidar com ausência de dados no momento em que uma tarefa é invocada. Isto permite que sejam tomadas medidas corretivas durante a execução do *workflow*, evitando erro e travamento da ação.

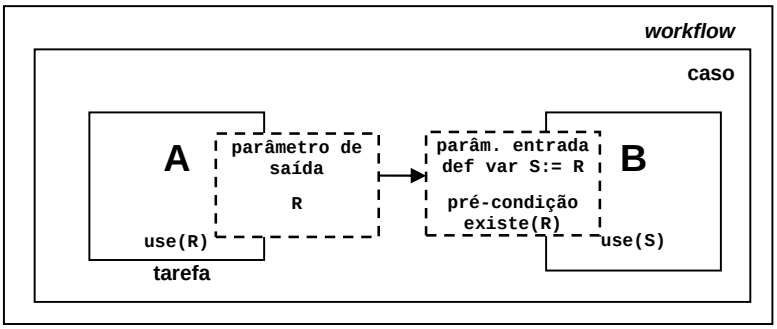



Figura 4.14. Pré-condição de tarefa – existência de dados [1]

**Implementação.** Geralmente, pré-condições de existência de dados são especificadas em parâmetros de entrada da tarefa no modelo de projeto do *workflow*. Nesse contexto, a existência de dados refere-se à capacidade de

determinar se um parâmetro requerido foi definido e fornecido à tarefa no momento da sua invocação da tarefa e se foi atribuído um valor. Uma das cinco ações é possível quando falta a identificação dos parâmetros:

1) Adiar início da tarefa até que os parâmetros necessários estejam disponíveis;
2) Especificar quais valores-padrão os parâmetros assumem quando indisponíveis;
3) Pedir valores interativamente para usuários do *workflow*;
4) Passar essa tarefa e acionar a seguinte tarefa (s);
5) Desativar essa linha de execução (*thread*).

***CPN Tools***: Utiliza a primeira abordagem, pois uma transição só é disparada se as fichas nos lugares de entrada satisfizerem as condições (expressões) dos arcos de entrada. No exemplo da Figura 4.15, apenas a transição *enviaEngradado* está habilitada para disparo, pois as fichas no lugar de entrada *Disponiveis* (10`Eng) satisfazem a condição da expressão de arco (*Eng*).

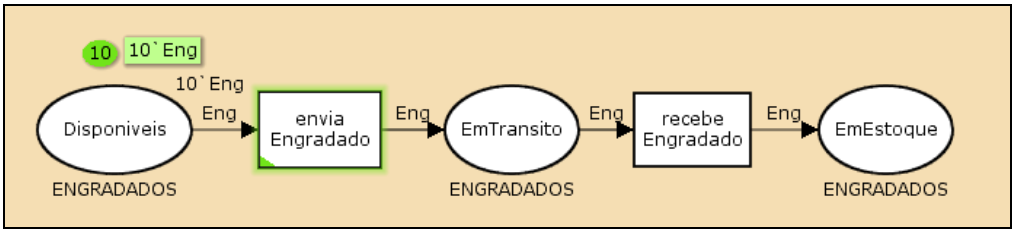



Figura 4.15. Pré-condição de tarefa – existência de dados em *CPN Tools*

Já a transição *recebeEngradado* está desabilitada para disparo, e seu início será adiado até que o lugar *emTransito* possua fichas satisfazendo a expressão de arco de entrada (*Eng*). No caso da Figura 4.16, ambas as transições estão habilitadas para disparo.

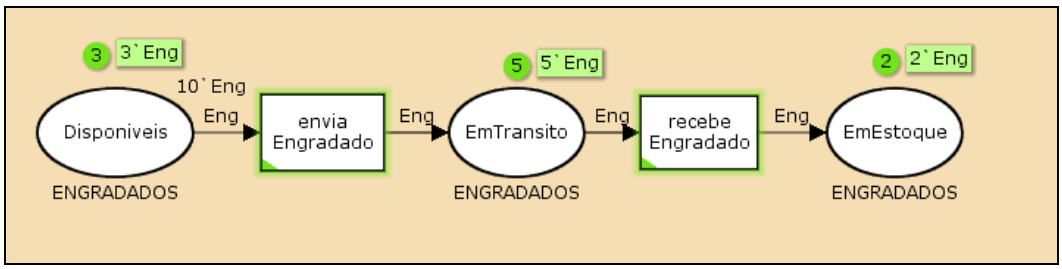



Figura 4.16. Pré-condição de tarefa – existência de dados em *CPN Tools*

**KEPLER**: A atividade só inicia quando os dados estão disponíveis nas portas de entrada do ator. Porém, os mecanismos de tratamento diferem com o diretor utilizado. Segue-se avaliação pra os diretores SDF e PN.

**Diretor SDF**. Nesse caso, a semântica de execução das atividades funciona assim: uma seqüência bem definida de realização das atividades é determinada, mas para que esse processamento ocorra, dados devem estar disponíveis nas portas de entrada de cada ator antes de sua execução. Na ausência de dados, a atividade é adiada até que os mesmos estejam disponíveis. Essa ordenação das atividades é feita automaticamente antes da execução do *workflow*.

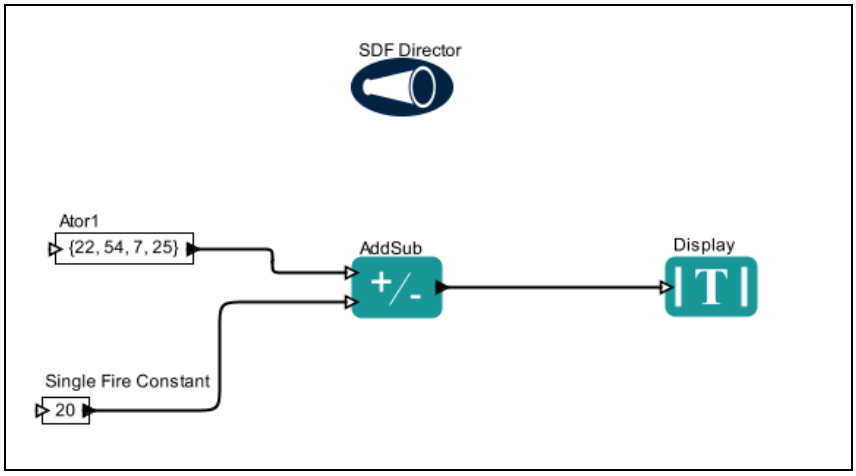



Figura 4.17. Pré-condição de tarefa – existência de dados – Sistema KEPLER com diretor SDF

**Diretor PN**. Nesse caso, todos os atores estão ativos simultaneamente, exceto aqueles sem todos os dados disponíveis na entrada. Um ator com mais de uma porta de entrada, como o *AddSub* na Figura 4.18, depende da existência de dados em todas as suas portas para então realizar sua tarefa. Assim, enquanto o ator aguarda a chegada de dados em uma de suas entradas, a(s) outra(s) pode(m) continuar recebendo dados e armazenando em uma fila, para utilizá-los posteriormente, quando a tarefa estiver apta para executar, ou seja, quando tiver dados em todas as suas portas de entrada. Porém, enquanto o ator aguarda chegada de dados em uma de suas portas, pode ocorrer estouro de fila nas demais, por excesso de dados. Isso ocasiona erro na execução do *workflow*.

A Figura 4.18 exemplifica o ator *AddSub* recebendo dados do *Ator1* várias vezes, porém, só uma vez os do *Single Fire Constant*. Assim, enquanto a atividade *AddSub* aguarda os dados na segunda porta de entrada, os dados chegando pela primeira porta são armazenados em fila (FIFO) até exceder a capacidade de armazenamento, gerando erro na execução do *workflow*.

Esse erro não ocorre no diretor SDF, pois não possui o conceito de fila para armazenar dados em suas portas de entrada. As atividades são ordenadas e realizadas para gerar a quantidade exata de dados esperada pela próxima atividade. Não se gerando o dado, o processamento do *workflow* se interrompe até que o dado esteja disponível.

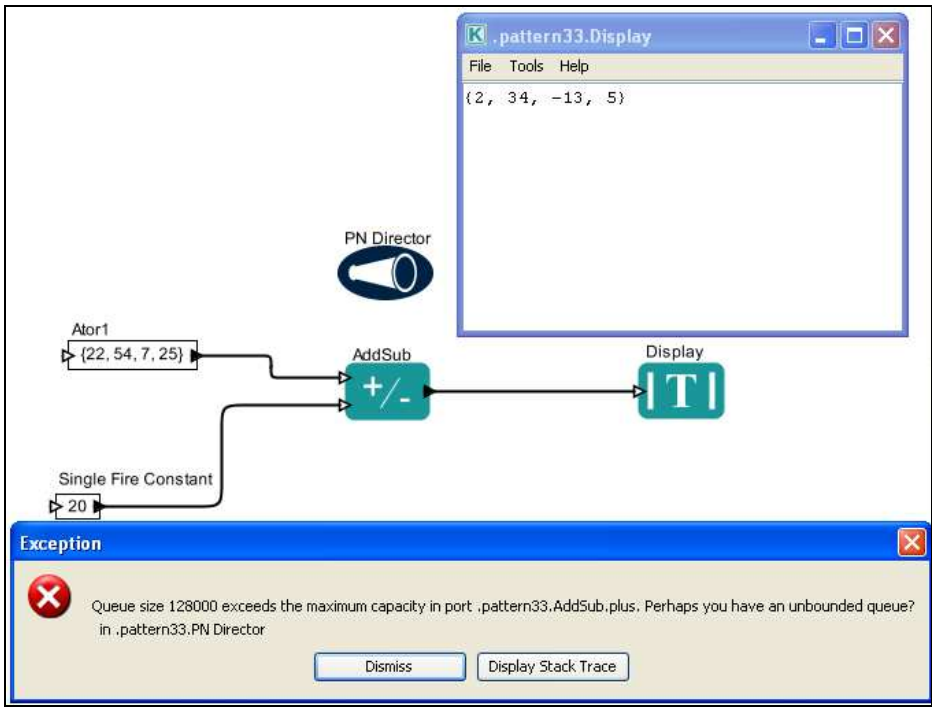



Figura 4.18. Pré-condição de tarefa – existência de dados – Sistema KEPLER com diretor PN

**Padrão 34: Pré-condição de tarefa – Valor de dados**

**Definição.** Pré-condições baseadas em dados podem ser especificadas para tarefas que têm como base o valor de determinado parâmetro no momento da execução.

**Motivação.** Capacidade de especificar pré-condições baseadas no valor dos parâmetros da tarefa, atrasando a execução da atividade (possivelmente indefinidamente), na qual uma pré-condição não é satisfeita.

**Implementação.** Existem três alternativas possíveis quando uma pré-condição baseada em valor não é encontrada:

1) A tarefa pode ser ignorada e as tarefas subseqüentes iniciadas;

2) O início da tarefa pode ser adiado até que a condição exigida seja satisfeita;

3) A linha de execução (*thread*) no *workflow* pode ser terminada.

***CPN Tools*:** implementa a segunda alternativa por meio de dois conceitos: expressão de arco e expressão de guarda. O primeiro, semelhante ao padrão anterior, permite o adiamento do disparo de uma transição até que a expressão, baseada em valores, associada ao arco de entrada dessa transição seja satisfeita. O segundo adia o disparo da transição enquanto uma condição associada à mesma, chamada expressão de guarda, não for satisfeita. Essa expressão é baseada em valores de dados do arco de entrada.

No exemplo da Figura 4.19, a transição *enviaCarta* não está habilitada, pois o valor da ficha em *VagasVazias* é zero, não satisfazendo a expressão de guarda *n>0*.

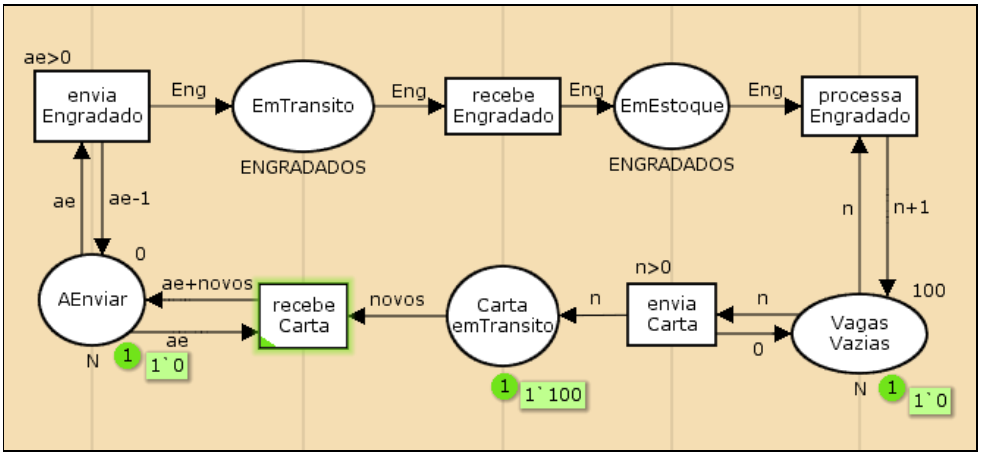



Figura 4.19. Pré-condição de tarefa – valor de dados – *CPN Tools*

**KEPLER**: não apresenta implementação direta de pré-condição nos atores com base em valores de dados. Esse tipo de comportamento é referenciado em [28] como característica do diretor *Rendezvous*, um modelo de computação baseado em CSP (*Communicating Sequencial Processes*) implementado no sistema PTOLEMY, predecessor de KEPLER. Porém, tal diretor não é implementado por esse sistema, desviando-se do escopo deste trabalho.

Por outro lado, para representar a necessidade de uma tarefa atender a uma pré-condição baseada em valores dos dados, é possível construir soluções a partir de alguns componentes do tipo *Workflow Control*, como o ator

*Boolean Switch*, usado no exemplo da Figura 4.20. O *subworkflow DemandaEstoque* só é ativado quando o parâmetro *Demanda* for diferente de zero.

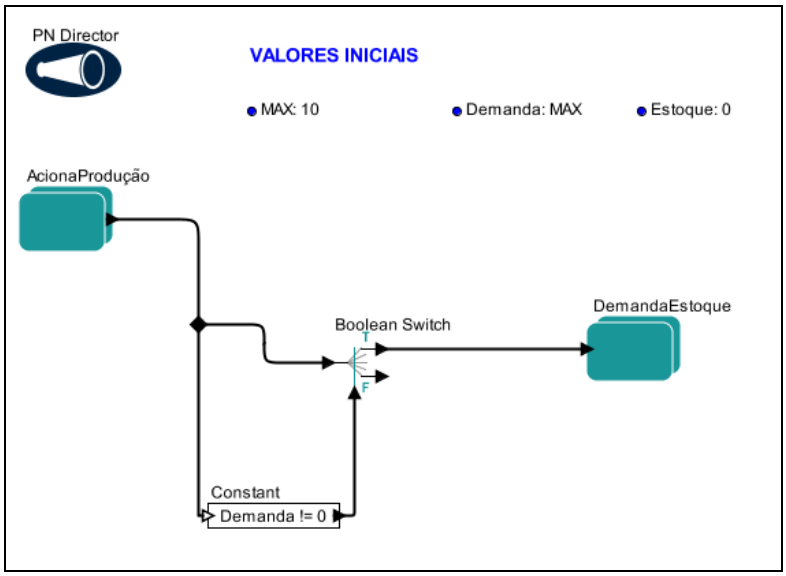



Figura 4.20. Pré-condição de tarefa – valor de dados – KEPLER

O *Boolean Switch* copia a ficha da sua porta de entrada (à esquerda) para a saída correspondente ao valor informado em sua porta de controle (embaixo). Ou seja, sempre que o valor na porta de controle for igual a *verdadeiro*, a saída *T* possui uma cópia da ficha da porta de entrada. Quando o valor de controle é *falso*, a saída *F* apresenta o valor copiado da entrada.

No exemplo da Figura 4.20, o valor gerado pelo ator *Aciona Produção* é enviado à porta de entrada do *Boolean Switch*, porém só é copiado para a entrada do ator *Demanda Estoque* quando o ator *Constant* gerar o valor *verdadeiro* em sua saída, ou seja, quando a parâmetro *Demanda* for diferente de 0.

**Padrão 35: Pós-condição de tarefa – Existência de dados (Figura 4.21)**

**Definição.** Pós-condições baseadas em dados podem ser especificadas para tarefas baseadas na existência de determinados parâmetros em sua saída no momento da execução.

**Motivação.** A implementação deste modelo garante que uma tarefa não pode concluir até que existam os parâmetros de saída especificados e tenham sido atribuídos valores a eles.

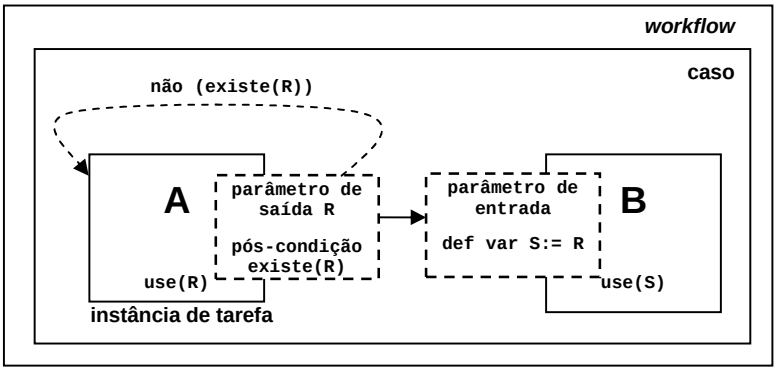



Figura 4.21. Pós-condição de tarefa – existência de dados [1]

**Implementação.** Existem duas alternativas para a implementação deste padrão em que tarefas têm uma pós-condição não atendida apesar de terem finalizado toda sua ação:

1) A tarefa poderá ser suspensa até que a pós-condição exigida seja satisfeita;

2) Implicitamente, a tarefa poderia ser repetida até que a pós-condição seja alcançada.

A implicação de ambos os cenários, no entanto, é que a tarefa não passa adiante seu controle de fluxo enquanto os parâmetros necessários não existirem e/ou não tiverem seus valores definidos.

***CPN Tools***: não implementa este padrão. Inexiste a suspensão de uma tarefa no modelo *CPN Tools*, ou repetição implícita, pela ausência de dados em sua saída. Os disparos das transições são controlados por pré-condições, ou seja, dependem das fichas em seus lugares de entrada.

**KEPLER**: não apresenta implementação de pós-condição com base em existência de dados. Semelhante ao padrão anterior, esse tipo de comportamento é referenciado em [28] como sendo uma característica do diretor *Rendezvous*, um modelo de computação implementado no Sistema PTOLEMY, porém, inexistente no KEPLER.

**Padrão 36: Pós-condição de tarefa – Valor de dados**

**Definição.** Pós-condições baseadas em dados podem ser especificadas para tarefas baseadas no valor de parâmetros específicos em suas saídas no momento da execução.

**Motivação.** A implementação desse padrão garantiria a não conclusão de uma tarefa até que parâmetros específicos de saída tenham um valor determinado ou que seus os dados estejam em um intervalo especificado.

**Implementação.** Similares ao Padrão 35, existem duas opções para o tratamento de conclusão de tarefa a partir de valores especificados de dados:

1) Atrasar a execução até que os valores necessários sejam alcançados.

2) Implicitamente reexecutar a tarefa.

***CPN Tools*:** de maneira semelhante ao Padrão 35, a *CPN Tools* não implementa esse padrão. Os disparos das transições são controlados por pré-condições, ou seja, dependem de fichas em seus lugares de entrada e não de saída.

**KEPLER:** semelhante ao Padrão 35, não apresenta mecanismos de implementação desse padrão.

**Padrão 37 – Gatilho (*trigger*) de tarefa baseado em evento**

**Definição.** Capacidade de um evento externo desencadear (iniciar) uma tarefa.

**Motivação.** Esse padrão concentra-se na possibilidade de um acontecimento externo acionar o início ou reinício de uma tarefa específica de uma instância de *workflow*. Isso permite que o controle de fluxo de tarefas seja influenciado por aplicações externas.

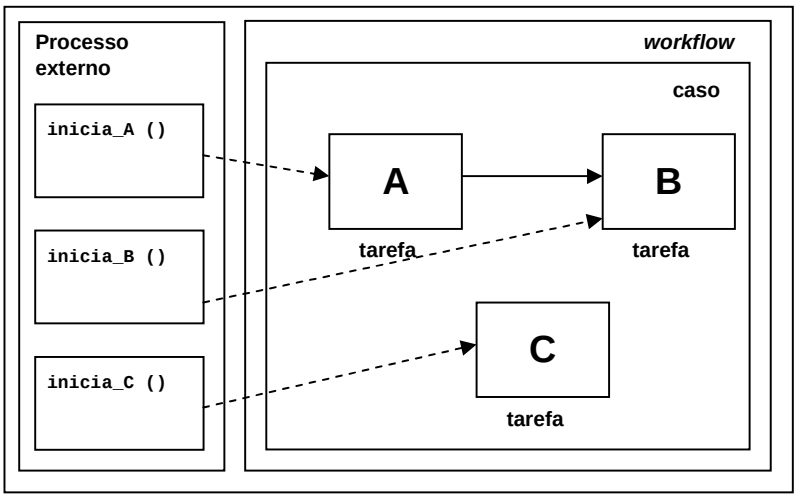



Figura 4.22. Gatilho de tarefa baseado em evento [1]

**Implementação.** Esse dispositivo geralmente assume a forma de uma interface externa do *workflow* que fornece um meio para aplicações externas acionarem a execução de uma instância específica da tarefa. Existem três diferentes cenários que podem surgir no contexto desse padrão (Figura 4.22).

A primeira alternativa (ilustrada pela função *inicia_A( )*) é a de que a instância da tarefa a ser iniciada é a primeira tarefa no *workflow*. Isso equivale, em termos de controle, a iniciar um novo caso de *workflow* em que A é a primeira tarefa.

A segunda alternativa (ilustrada pela função *inicia_B( )*), é a de que o evento externo está acionando o início de uma tarefa que está no meio de um processo de *workflow*. A tarefa já tem o controle passado para ela, mas sua execução é suspensa até a ocorrência de um evento externo. Tal situação é mostrada na Figura 4.20 com a tarefa B já desencadeada como resultado da conclusão de A, mas interrompida até que o evento *inicia_B( )* ocorra.

A terceira alternativa é a de que a tarefa é isolada do controle de fluxo principal no *workflow* e que o único modo em que pode ser iniciada é recebendo estímulo de um evento externo. A Figura 4.22 mostra instância da tarefa C que só pode ser acionada quando o evento *inicia_C ( )* é recebido.

***CPN Tools***: nenhuma das três situações é implementada pela *CPN Tools*, pois a ferramenta não dispõe de mecanismos de acionamento externo.

**KEPLER**: apesar de muitos atores da ferramenta KEPLER possuírem a possibilidade de acionamento por meio de um gatilho, a ferramenta não dispõe de mecanismos de acionamento externo.

**Padrão 38 – Gatilho de tarefa baseado em dados (Figura 4.23)**

**Definição.** Capacidade de tarefa específica ser desencadeada (iniciada) quando uma expressão baseada em dados do *workflow* é avaliada como *verdadeira.*

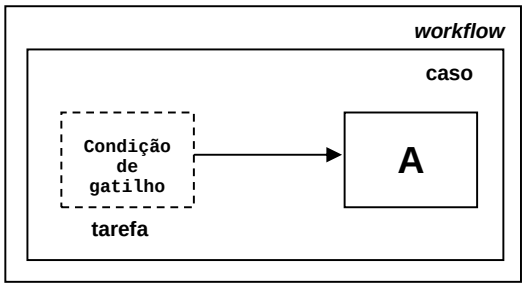



Figura 4.23. Gatilho de tarefa baseado em dados [1]

**Motivação.** Esse padrão fornece um meio de desencadear o início ou reinício de uma instância de tarefa, quando uma condição baseada em dados do *workflow* é satisfeita.

***CPN Tools***: Esse padrão é indiretamente implementado por meio de expressões de guarda incorporando condições de dados presentes no *workflow* ou nos arcos de entrada da tarefa a ser desencadeada. Esse mecanismo

habilita a transição para o disparo, porém, não o obriga. No exemplo da Figura 4.24, a tarefa *reabasteceePrateleira* só é habilitada para disparo quando o número de itens em *Prateleira* (*m*) fica abaixo do valor mínimo (*m<MIN*), mesmo com grande quantidade de produtos em *Estoque* (*n = 100*).

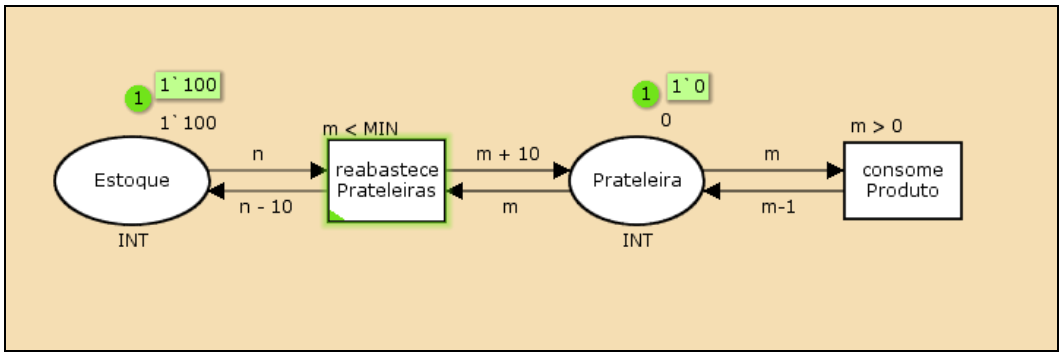



Figura 4.24. Gatilho de tarefa baseado em evento

Após reabastecimento da prateleira, a ação fica desabilitada até a quantidade de produtos na prateleira (*m*) recuperar o valor mínimo, após consumo de alguns itens (*consomeProduto*) (Figura 4.25).

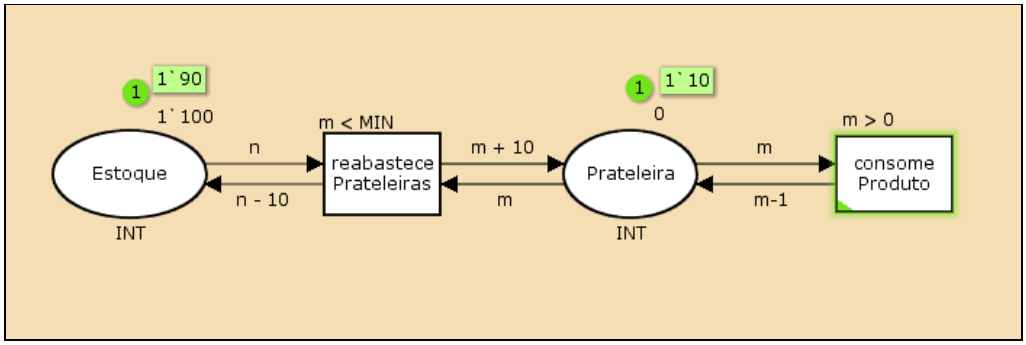



Figura 4.25. Após reabastecimento

A quantidade de produtos abaixo do mínimo habilita novamente a tarefa *reabasteceePrateleira* (Figura 4.26).

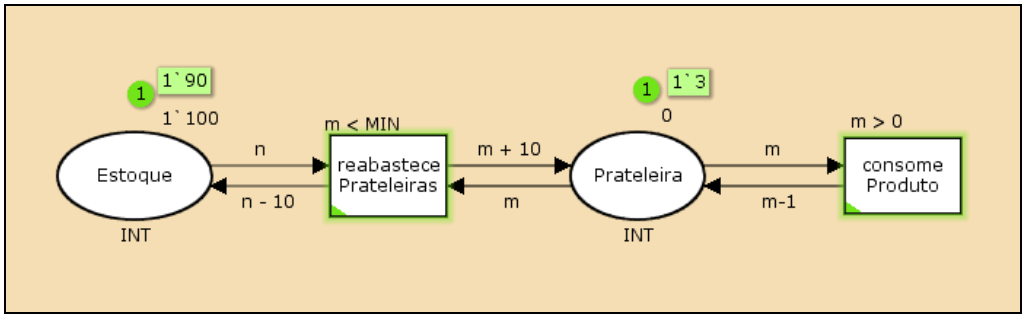



Figura 4.26. Quantidade de itens abaixo do valor mínimo

**KEPLER**: Vários atores da ferramenta KEPLER possuem porta de entrada definida como *trigger*, que dependem apenas da existência de fichas de dados nesta porta para serem acionados, não do valor das fichas existentes. Porém, é possível implementar esse padrão com o auxílio de outros atores, representando a expressão a ser avaliada como *verdadeira* ou *falsa*.

No exemplo da Figura 4.27 um ator *Uniform Distribution Random Number Generator* depende do acionamento de gatilho para gerar uma seqüência aleatória de valores uniformemente distribuídos entre os limites inferior e superior definidos. A cada iteração um novo número aleatório é gerado. A configuração desses limites pode ser feita por meio de parâmetros ou por fichas de dados com valores superior e inferior nas respectivas portas de entrada. Neste exemplo, foram configurados os parâmetros de limite inferior e superior iguais a 0.0 e 5.0, respectivamente.

O gatilho é disparado a partir da saída do ator *Boolean Switch* que copia a ficha de sua porta de entrada para a saída correspondente ao valor informado em sua porta de controle. Ou seja, se o valor na porta de controle for igual a *verdadeiro*, a saída *T* possui cópia da ficha da porta de entrada. Caso contrário, a saída *F* apresenta o valor copiado da entrada.

Para gerarmos o valor *verdadeiro* ou *falso* na entrada de controle do *Boolean Switch*, usou-se um componente de comparação chamado *Comparator* que, neste exemplo, compara os valores gerados por Ramp com o valor 5. Permanecendo os valores gerados menores ou iguais a 5, a saída é *falso*. Para valores maiores que 5, a saída é *verdadeiro*, acionando o componente *Boolean Switch* que, por sua vez, aciona o gatilho de *Uniform Distribution Random Number Generator*, gerando os valores apresentados pelo *DisplayOut* da Figura 4.25. O exemplo apresenta dez iterações, e os respectivos valores gerados pelos componentes *Ramp* e *Comparator* são exibidos pelos componentes *DisplayRamp* e *DisplayComparator*.

A partir desse exemplo pode-se afirmar que é possível representar acionamento de gatilho baseado em dados, porém, não de maneira direta, mas sempre com o auxilio de outros componentes da ferramenta. Alguns desses componentes (atores) são: *Comparator*, usado no exemplo apresentado; *Logic Function*, que envolve operações lógicas (*and*, *or*, *xor*, etc.); *Equals*, para operação de igualdade; e *IsPresent*, que verifica a presença de fichas em sua entrada, e devolve *verdadeiro* ou *falso* em sua saída.

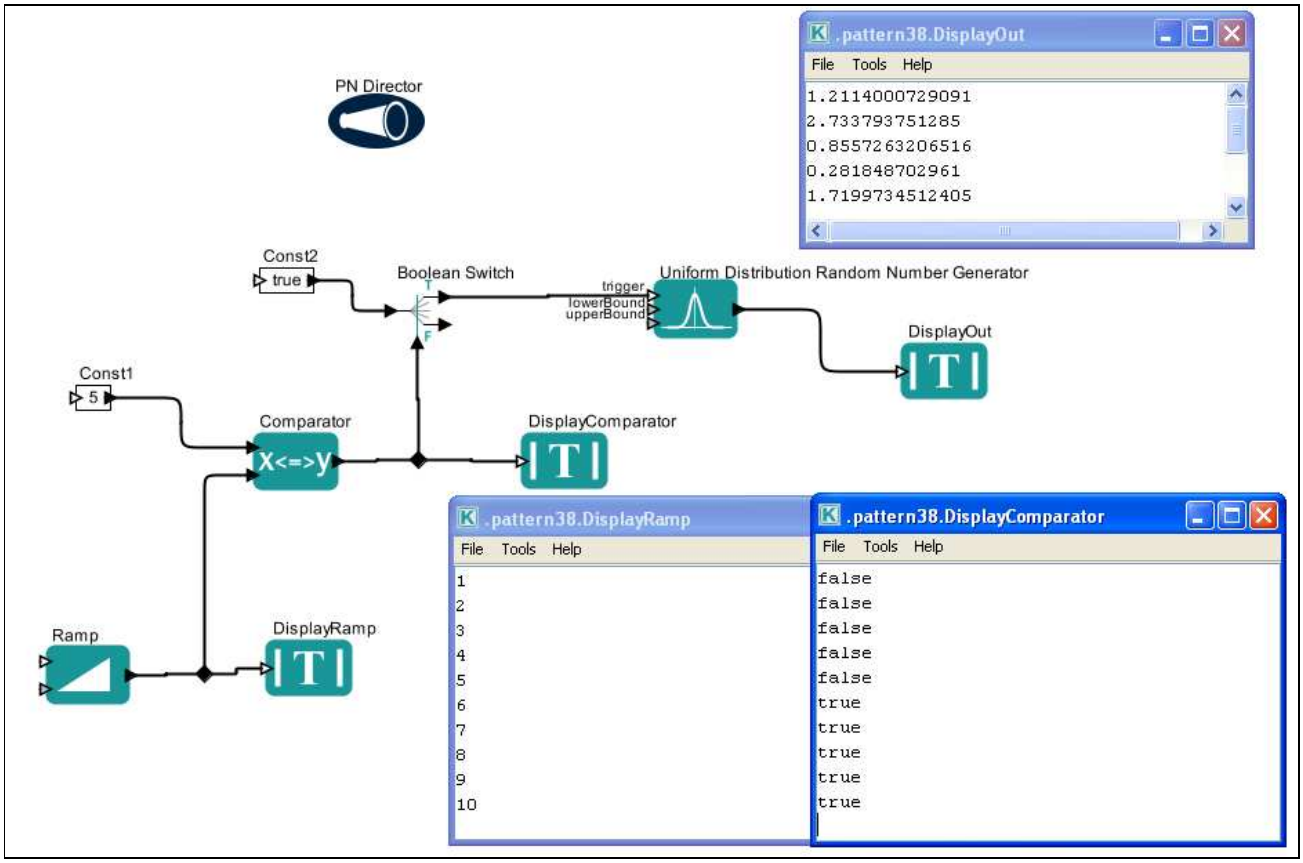



Figura 4.27. Gatilho de tarefa baseado em dados

**Padrão 39: Roteamento baseado em dados (Figura 4.28)**

**Definição.** Capacidade de alteração de controle de fluxo de tarefas em um caso de *workflow* como conseqüência da avaliação de uma expressão baseada em dados.

**Motivação.** Sem o conceito de roteamento baseado em dados, não seria possível para um *workflow* alterar o seu funcionamento, em resposta aos dados processados. Sua função seria rigidamente fixada em tempo de projeto.

Roteamento baseado em dados fornece a capacidade para a conclusão de uma instância de tarefa acionar uma ou várias instâncias de tarefas subseqüentes, dependendo do resultado de uma expressão baseada nos vários valores de dados do *workflow*.

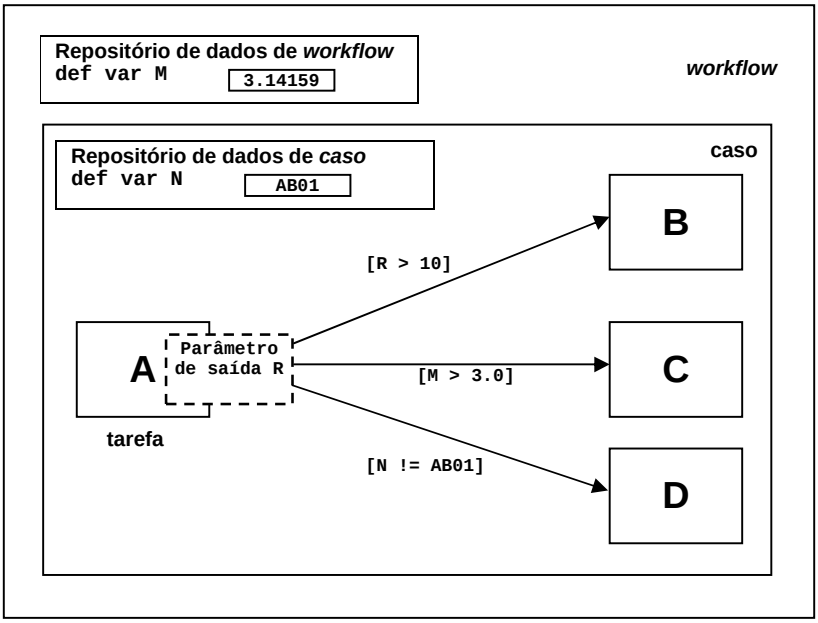



Figura 4.28. Roteamento baseado em dados [1]

**Implementação.** Este padrão serve como uma agregação de dois grandes padrões de controle de fluxo de tarefas [2] que dependem de dados do *workflow*:

1) Escolha exclusiva: o controle de fluxo de tarefas é passado para uma das várias instâncias de tarefa posteriores, dependendo do resultado de uma decisão ou o valor de uma expressão com base em dados do *workflow*;

2) Escolha múltipla: dependendo do resultado de uma decisão ou do valor de uma expressão com base nos dados do *workflow*, o controle de fluxo de tarefas é passado para uma ou mais instâncias de tarefas posteriores;

***CPN Tools*:** Ambos os padrões são implementados por meio de expressões de arco, como mostra o exemplo da Figura 4.29:

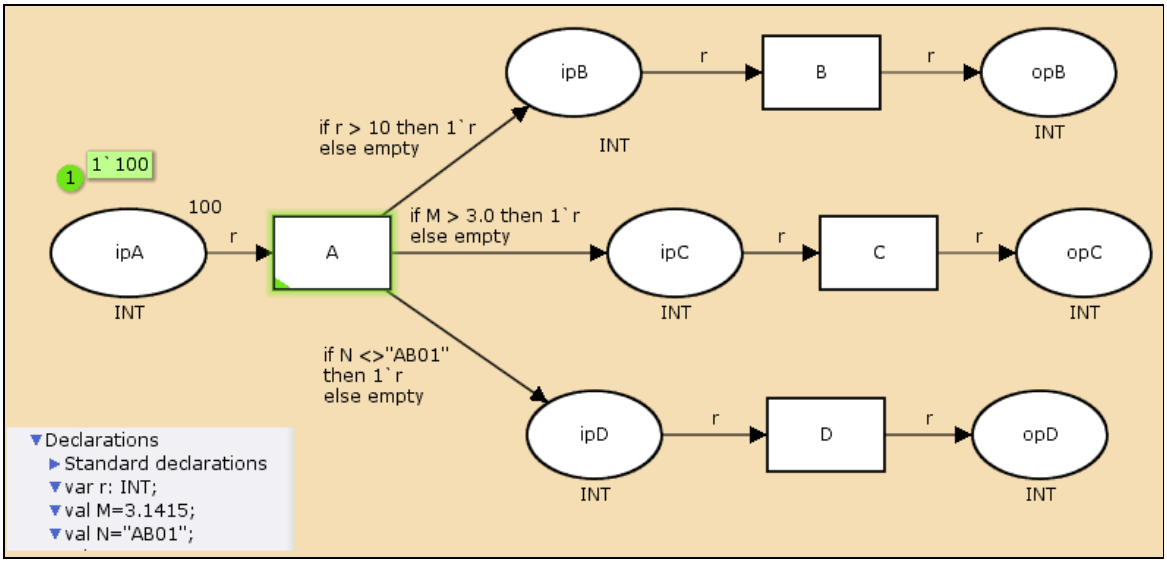



Figura 4.29. Roteamento baseado em dados - *CPN Tools*

**KEPLER:** Semelhante ao Padrão 38, o sistema KEPLER não permite implementar a alteração do controle de fluxo de tarefas de maneira direta, apenas por meio da combinação de componentes *workflow control*, conforme Figura 4.30.

Para implementação deste exemplo, usou-se um ator *Discrete Random Number Generator* para gerar números aleatórios entre 0 e 18. Os valores gerados nesta simulação aparecem no *DisplayRandom*. O componente *Boolean Switch1* só permite o envio da ficha fornecida pelo gerador para sua saída *T* quando seu valor é maior que dez. Essa avaliação é feita pelo *Comparator* e o resultado das várias iterações aparece no *DisplayFluxo1*.

O componente *Boolean Switch2* copia a ficha de sua entrada para a saída *T* sempre que *M* é maior que 3. O valor de *M* é fixo igual a 3.1415, gerando como saída para esse fluxo os mesmos valores fornecidos pelo gerador de números aleatórios. O resultado pode ser conferido no *DisplayFluxo2*.

Já o componente *Boolean Switch3*, só gera valor em sua saída *T* quando *N* é diferente de "*AB01*". Como essa situação nunca ocorre, vê-se em *DisplayFluxo3* que esse ator nunca dispara.

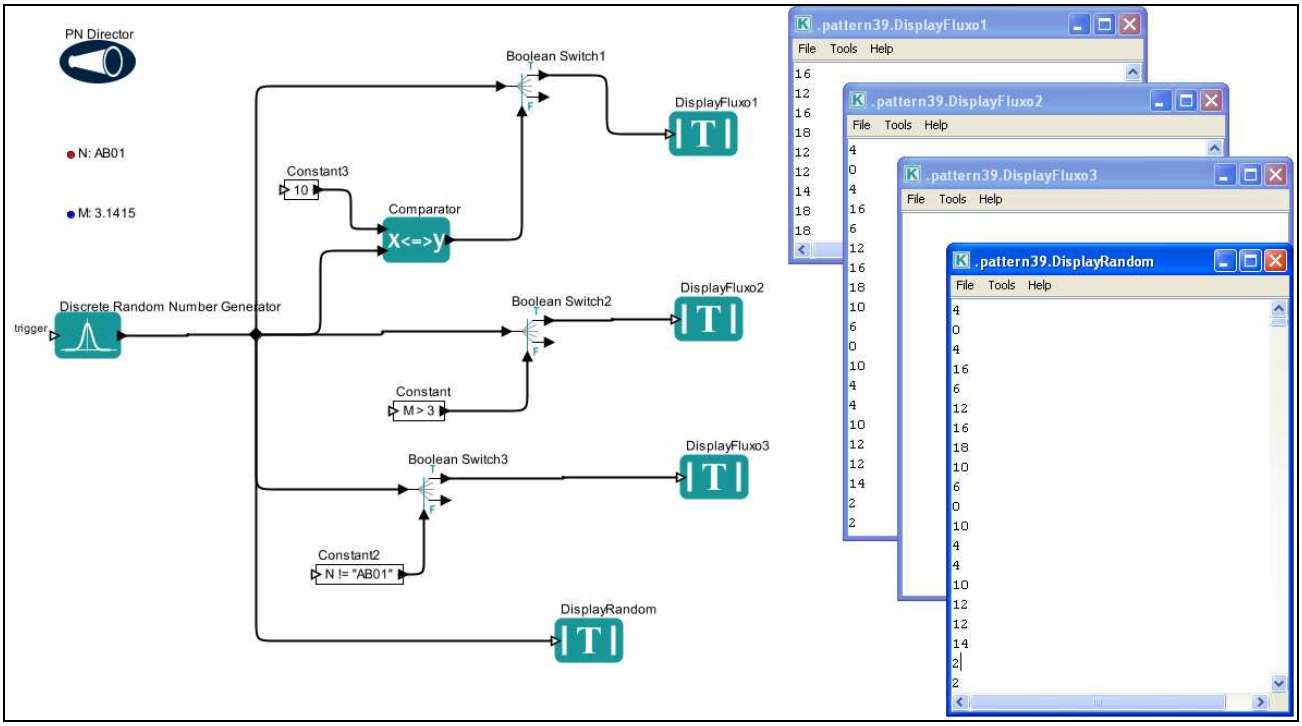



Figura 4.30. Roteamento baseado em dados - sistema KEPLER

## 4.3. Conclusão

Dadas as análises individuais dos padrões de controle de dados, as tabelas desta seção resumem as comparações apresentadas. A Tabela 4.1 refere-se aos padrões do grupo II – padrões de interação de dados internos do *workflow*, e a Tabela 4.2 refere-se aos padrões do grupo IV – padrões de roteamento baseado em dados. Nelas, a seguinte legenda será usada:

- "+" : ferramenta representa o padrão em questão;
- "–" : ferramenta não representa o padrão em questão;
- "+ / - " : ferramenta representa o padrão em questão de forma indireta ou parcial.

Observando a Tabela 4.1, nota-se que ambas as ferramentas mostram-se apropriadas para a representação de padrões que tratam a comunicação entre elementos ativos do *workflow*. Em redes de Petri coloridas, essa representação mostra-se fácil de implementar devido à existência de fichas de dados de diferentes tipos

definidos. Em grafos de *workflow*, a comunicação entre atores por meio de fichas de dados é uma de suas principais características, sendo fácil representar os padrões deste grupo.

| **No** | **Padrão** | **Definição** | ***CPN Tools*** | **KEPLER** |
|---|---|---|---|---|
| 8 | Tarefa a tarefa | Capacidade de comunicar dados entre uma instância de tarefa e outra em um mesmo caso. | + | + |
| 9 | Bloco de tarefas para *subworkflow* | Capacidade de transmitir dados a partir de uma instância de bloco de tarefas (ator composto ou super nodo) ao *subworkflow* correspondente, que define sua implementação. | + | + |
| 10 | *Subworkflow* para bloco de tarefas | Capacidade de transmitir dados a partir de um *subworkflow* de volta para seu bloco de tarefa (instância) correspondente. | + | + |

Tabela 4.1. Padrões de interação de dados internos do *workflow* (grupo II)

Porém, observando a Tabela 4.2, é possível perceber que a ferramenta *CPN Tools* mostra-se mais adequada para a representação dos padrões que caracterizam a maneira pela qual os dados podem influenciar o funcionamento do fluxo de trabalho (controle de fluxo de tarefas). Da análise feita dos padrões representados, é possível apontar fatores que contribuem para essa desigualdade da capacidade representativa:

- Implementação complexa de expressões de controle de fluxo de tarefas no Sistema KEPLER:

A maneira como o KEPLER representa decisões baseadas em dados não é tão simples como em *CPN Tools*. Muitas vezes, existe a necessidade de combinar diferentes atores de controle de fluxo para representar uma expressão baseada nos dados do modelo.

- Componentes de controle de fluxo não utilizados com diretor SDF:

O diretor SDF é estático, ou seja, antes do *workflow* iniciar sua execução, o diretor define a ordem da seqüência de atividades a serem realizadas. Isso não permite o uso de estruturas de controle dinâmicas, que decidem em tempo de execução as tarefas que devem ser realizadas e a quantidade de fichas a serem consumidas.

Os padrões não representados por nenhuma das ferramentas comparadas justificam-se por dois motivos:

- Não existe o conceito de uma atividade depender de uma pós-condição para ser realizada;
- A ausência de mecanismos para acionamento de tarefas a partir de eventos externos.

Detalhes da implementação dos padrões apresentados neste capítulo podem ser encontrados em: http://www.ime.usp.br/~grace/padroes.

| **No** | **Padrão** | **Definição** | ***CPN Tools*** | **KEPLER** |
|---|---|---|---|---|
| 33 | Pré-condição de tarefa – Existência de dados | pré-condições com base em dados podem ser especificadas por tarefas baseadas na presença de dados no momento da execução. | + | + |
| 34 | Pré-condição de tarefa – Valor de dados | pré-condições com base em dados podem ser especificadas por tarefas baseadas no valor de parâmetros específicos no momento da execução. | + | + / - |
| 35 | Pós-condição de tarefa – Existência de dados | pós-condições com base em dados podem ser especificadas por tarefas que se baseiam na existência de determinados parâmetros no momento da execução | - | - |
| 36 | Pós-condição de tarefa – Valor de dados | pós-condições com base em dados podem ser especificadas por tarefas que se baseiam no valor dos parâmetros específicos no momento da execução. | - | - |
| 37 | Gatilho (*trigger*) de tarefa baseado em evento | capacidade de um evento externo desencadear (iniciar) uma tarefa. | - | - |
| 38 | Gatilho de tarefa baseado em dados | capacidade de desencadear (iniciar) uma tarefa específica quando uma expressão baseada em dados do *workflow* é avaliada como *verdadeira.* | + / - | + / - |
| 39 | Roteamento baseado em dados | capacidade de alterar o controle de fluxo de tarefas em um caso de *workflow* como conseqüência da avaliação de uma expressão baseada em dados. | + | + / - |

Tabela 4.2. Padrões de roteamento baseado em dados (controle de fluxo de tarefas) (grupo IV)

**Capítulo 5**

# Conclusão

Este trabalho apresenta um estudo comparativo de controle e representação de dados em processos de negócios e *workflows* científicos tendo como base padrões de controle de dados. Para sua realização, foram detalhados dois formalismos: redes de Petri coloridas e grafos de *workflow* orientados a ator e suas respectivas implementações práticas: *CPN Tools* e KEPLER. Essa comparação permitiu identificar a abrangência e os limites de cada abordagem quanto aos aspectos de representação e controle de dados.

As tabelas 4.1 e 4.2 evidenciam algumas limitações de grafos de *workflows* quanto ao controle de fluxo de dados. Já *CPN Tools* não apresenta dificuldades quanto à representação e controle de dados.

A compreensão dos limites de cada abordagem é fundamental para o projeto e desenvolvimento de *workflows* científicos e processos de negócio. Muitas das ferramentas e iniciativas metodológicas disponíveis não valorizam a importância da representação dos padrões de controle de dados, gerando projetos incompletos, confusos e de difícil manutenção.

## 5.1. Principais contribuições

Como principais contribuições deste trabalho, pode-se citar:

- Caracterização de limites das redes de Petri colorida para representação de padrões de dados. Esses limites são: a) ausência de acionamento por evento externo ao modelo; b) não existência do conceito suspensão de tarefas tendo como base uma pós-condição.

- Caracterização de limites de grafos de *workflow* orientado a atores para representação de padrões de dados: além da ausência de acionamento por evento externo, e da não existência de pós-condição que suspenda a realização de uma tarefa, os grafos de *workflow* apresentam dificuldade de representação de estruturas dinâmicas. Em geral, é necessária a composição de vários atores para representarmos decisões baseadas em dados.
- Representação dos padrões de controle de dados de *workflow* em redes de Petri coloridas e grafos de *workflow*: alguns dos principais padrões de controle de dados apresentados por Russel e Aalst em [1] foram cobertos por implementações em *CPN Tools* e comparadas com implementações no sistema KEPLER. Detalhes dessas representações podem ser vistos no Capítulo 4.
- Um texto que sintetiza os fundamentos e principais elementos de redes de Petri coloridas, grafos de *workflow* e suas respectivas ferramentas: *CPN Tools* e KEPLER.

## 5.2. Sugestões para pesquisas futuras

Como atividades sugeridas para pesquisas futuras, incluem-se:

- Estudo, representação e implementação dos demais padrões de controle de dados usando *CPN Tools* e KEPLER;
- Estudo de formalismos algébricos, tais como LOTOS [35] e µCRL [36], para modelagem de processos de negócios que representam e controlam dados, comparando seu poder de representação em relação a redes de Petri coloridas, ratificando seus pontos positivos e identificando outras limitações;
- Estudo de alternativas para implementação de controle de dados usando a linguagem NPDL [37].

# Referências bibliográficas


[1] RUSSELL, N.; HOFSTEDE A.H.M. ter; EDMOND, D.; AALST, W.M.P. van der. *Workflow Data Patterns.* QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.

[2] AALST, W.M. P. van der; HOFSTEDE, A.H.M., KIEPUSZEWSKI, B.; BARROS, A.P. *Workflow Patterns.* Distributed and Parallel Databases, v.14(3), p. 5-51, Jul/2003.

[3] AALST, W.M. P. van der. *The Application of Petri Nets to Workflow Management*. The Journal of Circuits, Systems and Computers, v. 8, n. 1, p. 1-53. 1998.

[4] AALST, W.M. P. van der; HOFSTEDE, A.H.M. ter. *Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages*. In K. Jensen, editor, Proceedings of the Fourth Workshop on the Practical Use of Coloured Petri Nets and rdPc Tools (rdPc 2002), v. 560 of DAIMI, p. 1–20, Aarhus, Denmark, Ago/2002. University of Aarhus.

[5] AALST, W. M. P. van der; HEE, V. K. *Workflow management: models, methods and systems*. Cambridge: MIT Press, 2002.

[6] Sítio na Internet de padrões de *workflow*: www.*workflow*patterns.com (último acesso em 01/07/2008).

[7] WFMC. *Workflow Management Coalition Terminology and Glossary* (WFMC-TC-1011 Issue 3.0). Technical report. *Workflow* Management Coalition, 1999.

[8] SALIMIFARD, K.; WRIGHT, M. *Petri net-based modelling of workflow systems: An overview*. European Journal of Operational Research, 134 (3), pp. 664-676, 2001.

[9] MURATA, T. *Petri Nets: Properties, Analysis and Applications*. Proceedings of The IEEE, v. 77, n. 4, p. 541-580, abr. 1989.

[10] FOKKINK, W. *Introduction to Process Algebra. Secaucus*, NJ, EUA: Springer-Verlag New York, Inc., 2000. (Texts in Theoretical Computer Science (An EATCS Series)). ISBN 3-540-66579-X.

[11] MACIEL, P. R. M.; LINS, R. D.; CUNHA, P. R. F. *Introdução às Redes de Petri e Aplicações*. 10ª ESCOLA DE COMPUTAÇÃO, Campinas: Instituto de Computação, UNICAMP, 1996.

[12] PÁDUA, S. I. D.; SILVA, A. R. Y.; PORTO, A. J. V.; INAMASU, R. Y. *O potencial das redes de Petri em modelagem e análise de processos de negócios.* Gestão & Produção, São Carlos, v. 11, n. 1, p. 109-119, 2004.

[13] BRESSAN, G. *Modelagem e Simulação de Sistemas Computacionais - módulo Redes de Petri*; LARC-PCS/EPUSP, 2002. www.larc.usp.br/conteudo/universo/pcs012/modsim05.pdf (último acesso em 01/06/2007)

[14] JENSEN, K. *Coloured Petri Nets (2nd ed.): Basic concepts, Analysis Methods and Practical Use*: volume 1, Springer-Verlag, 1996.

[15] JENSEN, K. *An Introduction to the Practical Use of Coloured Petri Nets.* Lecture Notes in Computer Science: Lectures on Petri Nets II: Applications, v.1492, 1998.

[16] KRISTENSEN, L. M.; CHRISTENSEN, S.; JENSEN, K. *The practitioner's guide to Coloured Petri nets*. International Journal of *Software* Tools for Technology Transfer (STTT), v. 2, p. 98-132, 1998.

[17] Committee Draft ISO/IEC 15909. *High-level petri nets – concepts, definitions and graphical notation.* Oct 1997.Version 3.4.

[18] Sítio na Internet sobre ferramenta *Design/CPN*. http://www.daimi.au.dk/designCPN/ (último acesso em 01/02/2008).

[19] Sítio na Internet sobre ferramenta *CPN Tools*. http://wiki.daimi.au.dk/cpntools/ (último acesso em 01/02/2008).

[20] Sítio na Internet sobre Projeto *CPN2000*. http://www.daimi.au.dk/CPnets/CPN2000/ (último acesso em 01/02/2008).

[21] WESKE, M.; VOSSEN, G.; MEDEIROS, C. B. *Scientific Workflow Management: WASA Architecture and Applications*. Fachbericht Angewandte Mathematik und Informatik 03/96-I, Universitat Munster, 1996.

[22] ALTINTAS, Ilkay; BERKLEY, Chad; JAEGER, Efrat; JONES, Matthew; LUDASCHER , Bertram; MOCK, Steve. *KEPLER: An Extensible System for Design and Execution of Scientific Workflows.* In Proceedings of the 16th international Conference on Scientific and Statistical Database Management (SSDBM'04) - Volume 00, p.423, June 2004.

[23] LUDASCHER, B.; ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LEE, E.; TAO, J.; ZHAO, Y. *Scientific Workflow Management and the KEPLER System.* Concurrency and Computation: Practice and Experience, Issue 10, *Workflow* in Grid Systems, v. 18, p. 1039-1065, 2006.

[24] BOWERS, S.; LUDASCHER, B. *Actor-Oriented Design of Scientific Workflows*. Lecture Notes in Computer Science, v. 3716, p. 369-384, November 2005.

[25] BOWERS, S.; LUDASCHER, B.; NGU, A. H. H.; CRITCHLOW, T. *Enabling Scientific Workflow Reuse through Structured Composition of Dataflow and Control-Flow*. Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06) – p.70, 2006.

[26] OINN, T.; ADDIS, M.; FERRIS, J.; MARVIN, D.; SENGER, M.; GREENWOOD, M.; CARVER, T.; GLOVER, K.; POCOCK, M. R.; WIPAT, A.; LI, P. *Taverna: a tool for the composition and enactment of bioinformatics workflows.* Bioinformatics, v. 20, n.17, p. 3045-3054, 2004.

[27] MAJITHIA, S.; SHIELDS, M.; TAYLOR, I.; WANG, I. *Triana: A Graphical Web Service Composition and Execution Toolkit.* In Proc. of the IEEE Intl. Conf. on Web. Services (ICWS).IEEE Computer Society, 2004.

[28] Projeto e sistema PTOLEMY II. Department of EECS, UC Berkeley. http://PTOLEMY.eecs.berkeley.edu/PTOLEMYII/.

[29] Projeto e sistema DiscoveryNet. http://www.discovery-on-the.net/ (último acesso em 01/06/2007).

[30] Sistema Pipeline-Pilot da Scitegic. http://www.scitegic.com/ (último acesso em 01/06/2007).

[31] Sítio na Internet sobre sistema LabView da National Instruments. http://www.ni.com/labview/ (último acesso em 01/06/2007).

[32] LEE, E.A.; PARKS, T.M. *Dataflow process networks*. Proceedings of the IEEE, v. 83, n. 5, p. 773-801, 1995.

[33] AGHA, G. A. *Actors: A Model of Concurrent Computation in Distributed Systems*. Doctoral thesis, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 203 p., jun 1985.

[34] AGHA, G.; THATI, P. *An algebraic theory of actors and its application to a simple object-based language*. In Essays in Memory of Ole-Johan Dahl, v. 2635, pages 26-57, 2004.

[35] BOLOGNESI, T.; BRINKSMA, E. *Introduction to the ISO specification language LOTOS*. Comput. Netw. ISDN Syst., Elsevier Science Publishers B. V., Amsterdam, Holanda, v. 14, n. 1, p. 25–59, 1987. ISSN 0169-7552.

[36] GROOTE, J.F.; PONSE, A. *The syntax and semantics of μCRL.* In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen (editors), Algebra of Communicating Processes, Utrecht 1994. Workshops in Computing, p. 26-62, Springer-Verlag, 1995.

[37] BRAGHETTO, K. R. *Padrões de Fluxos de Processos em Banco de Dados Relacionais*. Dissertação (Mestrado) – Instituto de Matemática e Estatística da Universidade de São Paulo, 2006.