

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO



WORKFLOW NET POSSIBILÍSTICA PARA PROBLEMAS
DE NÃO CONFORMIDADE EM PROCESSOS DE
NEGÓCIOS

LEILIANE PEREIRA DE REZENDE

Uberlândia - Minas Gerais
2013

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO



LEILIANE PEREIRA DE REZENDE

WORKFLOW NET POSSIBILÍSTICA PARA PROBLEMAS DE NÃO CONFORMIDADE EM PROCESSOS DE NEGÓCIOS

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Stéphane Julia

Uberlândia - Minas Gerais

2013

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU

V111w Rezende, Leiliane Pereira de, 1988-
2013 Workflow net possibilística para problemas de não conformidade
em processos de negócios / Leiliane Pereira de Rezende. - 2012.
101 f. : il.

Orientador: Stéphane Julia.

Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia.

1. Computação - Teses. 2. Fluxo de trabalho - Teses. 2. Sistemas de informação gerencial - Teses. 3. Redes de petri - Teses. I. Julia, Stéphane. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**WorkFlow net possibilística para problemas de não conformidade em Processos de Negócios**” por **Leiliane Pereira de Rezende** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 28 de fevereiro de 2013

Orientador: _____

Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia

Banca Examinadora:

Prof. Dr. Paulo Eigi Miyagi
Universidade de São Paulo

Profa. Dra. Fernanda Francielle de Oliveira Malaquias
Universidade de Uberaba

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Data: 28 Fevereiro de 2013

Autor: **Leiliane Pereira de Rezende**
Título: **WorkFlow net possibilística para problemas de não conformidade em Processos de Negócios**
Faculdade: **Faculdade de Computação**
Grau: **Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

Autor

O AUTOR RESERVA PARA SI QUALQUER OUTRO DIREITO DE PUBLICAÇÃO DESTE DOCUMENTO, NÃO PODENDO O MESMO SER IMPRESSO OU REPRODUZIDO, SEJA NA TOTALIDADE OU EM PARTES, SEM A PERMISSÃO ESCRITA DO AUTOR.

“O Não é sempre certo na vida da gente. Por que não aprender a conquistar o Sim? O Sim é uma conquista.”

Eliana Zagui

Agradecimentos

Gostaria de agradecer ...

Primeiramente a Deus, pela graça da vida, por me amparar nos momentos difíceis, me dar força interior para superar as dificuldades, mostrar os caminhos nas horas incertas e me suprir em todas as minhas necessidades.

À minha mãe e minha irmã pela paciência dedicadas a mim nos momentos de angústia, pelo carinho, confiança, amizade, cuidado, apoio, e principalmente pelo amor, em todos os momentos da minha vida.

A meu pai e meu avô, que mesmo ausentes me ensinaram a ser forte e brigar pelo que desejo, me deram esperança de um mundo melhor e a consciência de auxiliar os menos favorecidos.

As minhas tias, tios, primas e primos pelos conselhos e por sempre me ampararem e apoiarem.

Aos meus amigos que distantes ou não eu sei que posso contar com eles.

A todos os professores que contribuíram pelo meu aprendizado, pela amizade e carinho a mim dados e pelo estímulo ao longo de todo o trabalho.

E, em especial ao professor Stéphane Julia, acima de tudo pelo profissionalismo, por acreditar em mim, me mostrar o caminho da ciência, pelos seus ensinamentos, paciência e compreensão desempenhados à mim ao longo desta jornada.

Aos que me ajudaram a ser quem sou, que depositam confiança em mim e para os quais sou uma esperança, resta-me afincadamente não vos desiludir.

A todos, o meu muito obrigada!

Resumo

As organizações contemporâneas usam a tecnologia da informação para apoiar as atividades e, possivelmente, automatizar os processos devido à sua complexidade e variedade. Os sistemas de Gerenciamento de Processos de Negócios influenciam significativamente na forma como as empresas organizam o trabalho, além de obrigá-las a ajustar seus processos de negócios a eles, propiciando a ocorrência de diversos problemas. Um dos principais é a incompatibilidade entre a forma preferida de trabalhar dos funcionários das empresas e a forma como o sistema de gerenciamento dos processos organiza o sequenciamento das atividades, forçando-os a trabalhar com muita frequência “fora do sistema”, executando processos de negócios “inadequados” em relação ao modelo definido pelo sistema de gerenciamento.

O objetivo deste trabalho é o de propor um novo modelo de processo de negócios baseado nas *WorkFlow nets* e nas “redes de Petri possibilísticas”. Os padrões de roteamento existentes nos processos de negócios são modelados pelas *WorkFlow nets* e, para expressar de uma forma mais realista a incerteza relacionada às atividades humanas, as “redes de Petri possibilísticas” com incerteza na marcação e no disparo das transições são consideradas. A abordagem proposta foca na flexibilidade por desvio na perspectiva do fluxo de controle e é capaz de lidar com as inconsistências, entre o modelo do processo e a execução do mesmo, geradas pelos atores humanos.

O processo de um “Serviço de Reclamação”, onde atores humanos são envolvidos nas atividades do processo, e dois exemplos de desvios originados a partir do fluxo de controle são utilizados para ilustrar a abordagem. As marcações e os disparos incertos permitem tratar os desvios de forma que nenhum bloqueio ocorra no estado do processo e o modelo não seja alterado.

A vantagem de tal abordagem é a manutenção das boas propriedades presentes no modelo do processo, pois a “rede de Petri clássica” não é modificada; logo, os métodos clássicos para análise de processos permanecem válidos. Além disso, através da defuzzificação do conhecimento, é possível formalizar o desvio por meio da sequência de atividades que não foram executadas, mas deveriam ter sido.

Palavras chave: Sistema de Gerenciamento de Processos de Negócios, processo de negócios, *WorkFlow net*, “rede de Petri possibilística”, flexibilidade por desvio na perspectiva do fluxo de controle, inconsistência, desvio, fator humano.

Abstract

Contemporary organizations use information technology to support activities and possibly automate processes due to its complexity and variety. Systems Business Process Management significantly influence the way companies organize work, besides forcing them to adjust their business processes to them, allowing the occurrence of various problems. One of the principal is the incompatibility between the preferred way of the work of company employees and the way the system process management organizes the sequencing of activities, forcing them to work very often “outside the system” by running business processes “inadequate” compared to the model set by the management system.

The objective of this work is to propose a new model of the business process based on WorkFlow nets and on “possibilistic Petri nets”. The routing patterns in existing business processes are modeled by WorkFlow nets, and to express in a more realistic way the uncertainty attached to human activities, “possibilistic Petri nets” with uncertainty on the marking and on the transition firing are considered. The proposed approach focuses on flexibility by deviation from the perspective of the control flow and is able to deal with the inconsistencies between the process model and the execution, generated by human actors.

The process “Service Complaint”, where human actors are involved in the process activities, and two examples of deviations arising from the control flow are used to illustrate the approach. The markings and firings uncertain allow deviate so that no blockage occurs in the state of the process and the model is not changed.

The advantage of this approach is the maintenance of good properties in the model of the process, because the “classical Petri nets” is not modified; therefore the classical methods for analyzing processes remain valid. Furthermore, through the defuzzification of knowledge, it is possible to formalize the deviation through the sequence of activities that have not been executed, but should have been.

Keywords: *System Business Process Management, business process, WorkFlow net, “possibilistic Petri net”, flexibility by deviation from the perspective of the control flow, inconsistency, human factor.*

Sumário

Lista de Figuras	xvii
1 Introdução	19
1.1 Contribuições	22
1.2 Organização da dissertação	23
2 Revisão Bibliográfica	25
2.1 Considerações Finais do Capítulo	31
3 Fundamentos Teóricos	33
3.1 Redes de Petri Clássica	33
3.2 Redes de Petri a Objetos	36
3.3 Redes de Petri Possibilísticas	40
3.3.1 Algoritmo de Defuzzificação	46
3.4 <i>Workflow nets</i>	52
3.4.1 Processos	53
3.4.2 Roteamentos	54
3.4.3 Acionamentos	55
3.4.4 Soundness	56
3.4.5 Monitoramento de Processos	56
3.5 Considerações Finais do Capítulo	59
4 <i>Workflow net</i> Possibilística	61
4.1 Cenário I - Desvio Fraco	68
4.2 Cenário II - Desvio Forte	73
4.3 Considerações Finais do Capítulo	86
5 Conclusão e Trabalhos Futuros	89
Referências Bibliográficas	93

Lista de Figuras

3.1	Exemplos de sensibilização e disparo de transição em uma rede de Petri.	34
3.2	Sequência de disparo de transições.	35
3.3	Exemplo de uma “rede de Petri a Objetos”.	38
3.4	Exemplo de disparo da transição t	40
3.5	Exemplo de “rede de Petri possibilística”.	44
3.6	Marcação.	45
3.7	Distribuição das possibilidades das localizações de o_1, o_2 e o_3	46
3.8	Exemplo de uma “rede de Petri possibilística” para uso do algoritmo de defuzzificação.	48
3.9	Aplicação do algoritmo de defuzzificação.	49
3.10	Continuação da aplicação do algoritmo de defuzzificação.	50
3.11	Disparo da transição t_3 , como na “rede de Petri clássica”, após a execução do algoritmo de defuzzificação.	52
3.12	Elementos de Modelagem de uma <i>Workflow net</i>	53
3.13	<i>Workflow net</i> para o processo de tratamento de reclamações e os seus acionamentos.	53
3.14	Bloco bem formado.	55
3.15	Tipos de tarefas nos acionamentos.	55
3.16	(a) <i>Workflow net</i> tradicional; (b) <i>Workflow net</i> com a execução explícita da tarefa; e (c) <i>Workflow net</i> com evento de cancelamento . . .	57
3.17	<i>Workflow net</i> com a execução explícita das tarefas.	58
4.1	“Jogador” de “rede de Petri clássica interpretada”.	62
4.2	“Serviço de Reclamações”.	66
4.3	Flexibilidade por desvio.	67
4.4	Flexibilidade por desvio na perspectiva do fluxo de controle.	67

4.5	Fração do Processo de “Serviço de Reclamações”	68
4.6	Resultados da simulação do cenário I considerando o “jogador” de “rede de Petri clássica interpretada”.	70
4.7	“Jogador” de “rede de Petri possibilística”.	71
4.8	Resultados da simulação do cenário I.	72
4.9	Distribuições das possibilidades relacionadas aos locais dos objetos c_{11} e c_{12}	73
4.10	Fração do processo “Serviço de Reclamações” alterado para suportar o Cenário I.	73
4.11	Estado inicial a ser considerado no cenário II.	75
4.12	Novo “jogador” de “rede de Petri possibilística”.	76
4.13	Resultados da simulação do cenário II.	80
4.14	Resultados da simulação do cenário II.	82
4.15	Resultados da simulação do cenário II.	83
4.16	O processo “Serviço de Reclamações” modificado para suportar o cenário II.	85
4.17	“Serviço de Reclamações” com a presença de novas transições.	87

Capítulo 1

Introdução

O termo *workflow* consiste na utilização de elementos computacionais para auxiliar a execução de processos [1]. *Workflows* são tipicamente usados para descrever a interação entre empregados e recursos materiais e computacionais para execução desses processos.

Os Sistemas de Gerenciamento de *WorkFlow*, do inglês *WorkFlow Management System* (WfMS), permitem a automação de processos que envolvem atividades humanas e computacionais, incluindo interação com ferramentas e aplicações auxiliares [1]. Nesses sistemas os processos são modelados em *workflows* que permitem a definição da sequência das atividades realizadas durante um processo, bem como os responsáveis e recursos de cada uma.

Segundo Aalst [2], a meta fundamental do gerenciamento de *workflow* é ter a certeza de que atividades corretas são executadas pelas pessoas certas e no tempo certo. Além disso, ele destaca que a área de Gerenciamento de Processos de Negócio, do inglês *Business Process Management* (BPM), tem recebido uma atenção considerável nos últimos anos em razão de seu potencial em aumentar significativamente a produtividade e economizar gastos [3].

Conforme Pádua et al. [4], um processo de negócio é um conjunto de atividades que podem, ou não, ser executadas simultaneamente, com alguma especificação de controle e fluxo de dados entre as atividades. Aalst et al. [3] destacam que o conceito de processo é fundamental e serve como um ponto de partida para o entendimento de como o negócio opera e quais oportunidades existem para coordenar suas atividades constituintes.

Empresas internacionais, hospitais, universidades, lojas on-line, empresas de pequeno porte, etc., todos eles envolvem um grande número de pessoas que realizam um amplo

número de processos. Estes processos tornam-se cada vez mais complexos e requerem um controle rígido para evitar erros ao mesmo tempo em que precisam de mais flexibilidade para lidar com situações inesperadas.

Apesar disso, os Sistemas de Gerenciamento de *Workflow* geralmente são desenvolvidos para suportar apenas a modelagem de processos de negócios rigidamente estruturados [3]. Logo, os usuários são forçados a revisar constantemente o modelo do processo e/ou trabalhar fora do sistema, realizando tarefas informais, a fim de completar suas tarefas com sucesso. Com isso a produtividade é reduzida, o tempo de processamento é maior e possíveis desvios e/ou inconsistências podem surgir.

Traduzir os conceitos abstratos e as descrições das práticas e regras de negócio em modelos de processos está longe de ser um exercício trivial. Mesmo para os processos altamente estruturados (tais como em ambientes médicos e bancários), torna-se difícil (ou mesmo impossível) capturar com sucesso todas as relações entre atividades, e em particular, todas as sequências possíveis de tarefas, num modelo de negócios sem que este se torne muito complexo.

O termo “flexibilidade” é geralmente usado para designar o grau ao qual um sistema é capaz de lidar ou suportar os desvios esperados ou não na execução das instâncias do processo, tanto dentro do contexto da mesma ou a partir do ambiente externo, sem impacto negativo sobre a essência do processo ou acerca de sua conclusão. Historicamente, a flexibilidade nos processos iniciou-se nos sistemas de manufatura [5, 6, 7, 8, 9, 10, 11], onde tentativas de se adaptar às situações inesperadas ou aos roteiros alternativos foram definidas. Apesar disso, no setor administrativo, a flexibilidade proporcionada pelos sistemas para acomodar a evolução natural do processo ou as metas organizacionais é insignificante ou nem existe [3]. Tal fato ocorre por causa destes processos dependerem, em muitos aspectos, dos funcionários humanos que nem sempre respeitam com muito rigor as regras de negócio da empresa onde trabalham.

Na verdade, é por causa das discrepâncias entre as atividades do mundo real e as representações formais das mesmas que as instâncias do processo de negócio, normalmente, apresentam suas exceções durante a execução. Tradicionalmente, as exceções são entendidas como sendo eventos que, por definição, ocorrem raramente. Entretanto, em um processo de negócio, uma exceção pode ser descrita como um evento que é considerado fora do comportamento “normal” de tal processo. Em vez de ser um erro, é simplesmente um evento que é considerado como sendo um desvio a partir do fluxo de controle esperado.

Num modelo de um processo de negócio se uma exceção é esperada, ou até mesmo poderia ter sido concebivelmente antecipada, o projetista tem então, *a priori*, o conhecimento de tal evento e deveria tê-la incorporado a ele. Mas, se uma exceção é inesperada, ou não poderia ter sido prevista, dado o âmbito do conhecimento disponível no momento da concepção, o modelo é considerado deficiente e necessita ser alterado para incluir este evento previamente inimaginável. Contudo, para alterar o modelo em tempo de execução, o andamento do processo deveria ser suspenso e uma intervenção manual deveria ser utilizada. Além disso, se todos os possíveis cenários de uma exceção são integrados *a priori* ao modelo a complexidade do mesmo se tornaria muito alta.

Desde meados dos anos noventa, muitas pesquisas têm sido realizadas sobre as questões relacionadas com a flexibilidade e com a manipulação das exceções nos Sistemas de Gerenciamento de *Workflow* [3]. Tais pesquisas foram iniciadas porque, geralmente, os sistemas comerciais de gerenciamento de *workflow* precisam ser totalmente modelados com todos os cenários possíveis definidos antes que possam ser instanciados, e que possíveis alterações devem ser incorporadas através da modificação do modelo estaticamente.

Atualmente, Zazworka et al. [12], entre outros autores, têm pesquisado sobre o problema de não conformidade presente nos processos decorrentes da falta de flexibilidade, porém a maioria dos modelos de processos realizam apenas a detecção do desvio e/ou da inconsistência a partir da comparação de um modelo de referência com uma instância do mesmo sendo executada.

Thompson e Torabi [13] definem a não conformidade como a diferença entre o processo definido e o executado, dividindo-a em duas sub-classes: desvios e inconsistências. Desvio refere-se a uma ação realizada que não está descrita no modelo do processo ou que viola algumas restrições. Inconsistência refere-se ao estado do processo resultante do desvio.

Diversos trabalhos, como os desenvolvidos por Aalst [14], consideram a teoria das redes de Petri [15] como uma ferramenta eficiente na modelagem e análise de processos de negócios. As *Workflow nets*, definidas por Aalst em [2, 14], são redes de Petri que modelam processos de negócios dentro de uma rígida estrutura. Em [16], uma extensão das *Workflow nets* é apresentada. Este modelo é chamado de *Time Workflow net* e associa intervalos de tempo às transições do modelo de rede de Petri correspondente.

Aalst [2] afirma que há várias razões para usar redes de Petri na modelagem de processos de negócios: semântica formal, natureza gráfica, expressividade, propriedades, técnicas de análise e o fato de ser fornecida de forma independente. Ele ainda acrescenta que o

uso de tal formalismo tem uma série de vantagens importantes, como o fato de prevenir ambiguidades e contradições em contraste com a maioria das técnicas de diagramação informais.

As redes de Petri *fuzzy*, uma combinação da teoria dos conjuntos nebulosos [17] com a teoria das redes de Petri [15], é uma ferramenta para representação do conhecimento incerto sobre um estado de um sistema. Desde 1988, com o trabalho de Looney [18], diversos autores, tanto da comunidade das redes de Petri quanto de Inteligência Artificial, têm proposto diferentes tipos de redes de Petri *fuzzy* [6, 19, 20]. Sob o mesmo nome, esses modelos são diferentes entre si. Suas diferenças estão relacionadas às ferramentas *fuzzy* utilizadas (lógica *fuzzy* [21] ou teoria da possibilidade [22]) e aos elementos que são fuzzificados (por exemplo: marcações, tempo de disparo, etc.). Dessa forma, possíveis inconsistências que acontecem durante a execução de um processo são permitidas evitando que um desvio do processo em relação a um modelo de referência se transforme numa inconsistência, porém tais abordagens foram definidas no contexto da manufatura [23].

A proposta deste trabalho é formalizar, nos processos de negócios, possíveis desvios humanos utilizando a teoria da possibilidade para garantir uma maior flexibilidade na execução das tarefas por parte dos atores envolvidos.

Para isso, pretende-se combinar as *WorkFlow nets* com a teoria das possibilidades e definir um novo modelo de *WorkFlow net* possibilística que, através da execução de um “jogador” de rede de Petri, poderá monitorar em tempo real processos de negócios flexíveis onde decisões humanas não completamente previstas no modelo não tornarão necessariamente inconsistente o estado do sistema de gerenciamento.

1.1 Contribuições

Este trabalho apresenta 4 contribuições principais:

- definição de um novo modelo baseado nas *WorkFlow nets* de Aalst [14] e nas *Fuzzy Petri Net* de Cardoso [20] que seja consistente com a execução dos processos de negócios cujas partes das atividades sejam realizadas por atores humanos que nem sempre respeitam os processos definidos dentro de uma organização específica;
- definição de uma arquitetura baseada num mecanismo de inferência especializado, chamado de “Jogador” de “rede de Petri possibilística”, cuja funcionalidade é o

monitoramento em tempo real de processos de negócios existentes em estruturas organizacionais onde partes das atividades são realizadas por funcionários (atores humanos);

- tratamento de certos tipos de desvios e/ou inconsistências nos processos de negócios gerados pelos atores humanos onde o fluxo de controle não suporta tais eventos, utilizando a marcação incerta e os disparos das redes de Petri possibilísticas;
- registro das sequências de disparo incerto possibilitando o diagnóstico dos problemas entre o modelo definido e o executado (o que não aconteceu, mas deveria ter acontecido). Esta sequência não é obtida numa rede de Petri com diversos cenários definidos, ou seja, com várias alternativas de execução, impossibilitando assim a representação formal do desvio.

1.2 Organização da dissertação

Esta dissertação está dividida em 5 capítulos, organizados da forma como se segue.

No capítulo 2 a revisão bibliográfica é apresentada. Nele, as principais contribuições existentes na literatura abordando o tema de “não conformidade nos processos de negócios” são descritas juntamente com as limitações existentes em cada proposta apresentada.

No capítulo 3, as fundamentações teóricas necessárias para o entendimento deste trabalho são apresentadas. Na Seção 3.1 faz-se uma introdução sobre o que são as redes de Petri clássica, seus elementos, as regras de sensibilização e disparo de uma transição e a equação fundamental. Nas Seções 3.2 e 3.3 as redes de Petri a Objeto e possibilísticas são definidas respectivamente. Por fim, as *WorkFlow nets* são definidas na Seção 3.4, apresentando seus elementos de modelagem, a definição de processos, roteamentos e acionamentos, a propriedade *soundness* e o monitoramento de processos.

No capítulo 4, o modelo proposto é apresentado definindo a *WorkFlow net* possibilística. Nas Seções 4.1 e 4.2 descreve-se situações de inconsistências geradas em tempo de execução no processo e como são tratadas de forma a evitar possíveis bloqueios no processo. Em cada seção, o “jogador” de “rede de Petri possibilística” é alterado para que em cada caso o processo possa continuar sua execução sem grandes transtornos.

Finalmente, o Capítulo 5 apresenta a conclusão deste trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

A importância da flexibilidade nos Sistemas de Gerenciamento de *Workflow* tem sido reconhecida por muitos pesquisadores [24, 25]. O principal problema em relação à flexibilidade é a necessidade de especificar os processos de negócios em detalhe. Entretanto, estes processos não são previstos com uma precisão elevada [24, 26] e necessitam de adaptações constantes para se ajustarem aos ambientes em transformação [26].

Com base em experiências práticas, Reijers [25] fornece uma discussão sobre o fato de que os Sistemas de Gerenciamento de *Workflow* não conseguiram trazer a flexibilidade pretendida por extrair a noção da lógica de coordenação dos processos de negócios a partir das aplicações. Além disso, Reijers argumenta que, em vez de flexibilidade, eles melhoraram os aspectos logísticos do trabalho: o tempo de execução foi reduzido beneficiando os gestores e todos os dados relevantes ao processo são disponíveis automaticamente auxiliando os trabalhadores além de orientá-los durante a execução.

Em [24], Heinl et al. abordaram a questão da flexibilidade num estudo de caso realizado em uma grande empresa de pesquisa de mercado que utiliza os Sistemas de Gerenciamento de *Workflow* para apoiar mais de 400 processos. Este estudo mostrou que os problemas surgem devido ao fato de que é difícil prever todas as alternativas de execução dos processos de negócios ao especificar um modelo de processo, e que a flexibilidade na execução das instâncias dos modelos de processos é necessária para lidar com estes problemas. A flexibilidade de um Sistema de Gerenciamento de *Workflow* é vista como o grau em que os usuários podem escolher entre várias alternativas durante a execução dos modelos do processo. A partir disso, os autores identificaram dois

conceitos que devem ser suportados por um Sistema de Gerenciamento de *Workflow* flexível: flexibilidade por seleção¹ e flexibilidade por adaptação².

Flexibilidade por seleção dá ao usuário um certo grau de liberdade, oferecendo múltiplas alternativas de execução. Flexibilidade por adaptação considera a adição de uma ou mais alternativas de execução inesperadas para um modelo de processo enquanto o modelo está sendo executado, sendo relevante quando uma instância do *workflow* é detectada como sendo incompleta ou errônea durante a execução.

Contudo, esses dois tipos de flexibilidade apresentam algumas limitações. A flexibilidade por seleção deve ser antecipada e incluída (direta ou indiretamente) dentro da especificação do *workflow*. E, para suportar a flexibilidade por adaptação, o Sistema de Gerenciamento de *Workflow* tem que prover uma funcionalidade adicional e ferramentas para trocar o tipo do *workflow* e integrar essas trocas durante a execução.

Schonenberg et al., em [27, 28], propuseram quatro tipos de flexibilidades: flexibilidade de *design*³, flexibilidade por subespecificação⁴, flexibilidade por mudança⁵ e flexibilidade por desvio⁶.

Flexibilidade de *design* é a habilidade de incluir múltiplos cenários no modelo do processo. Flexibilidade por subespecificação é a capacidade de deixar partes do modelo do processo não especificada. Flexibilidade por mudança é a habilidade de modificar o modelo do processo em tempo de execução. E, flexibilidade por desvio é a capacidade de desviar das alternativas de execução especificadas no modelo de processo sem alterá-lo.

Snowdon et al. identificam três fatores que motivam a necessidade de diferentes tipos de flexibilidade [26]. Em primeiro lugar, a necessidade de flexibilidade nos tipos surge a partir da variedade de diferentes sistemas de informação. Em segundo lugar, a flexibilidade de volume por lidar com uma grande quantidade de tipos de informação e, em terceiro lugar, a flexibilidade estrutural devido à necessidade de trabalhar em diferentes modos.

A teoria das redes de Petri [15, 29] é considerada em muitos trabalhos como uma ferramenta eficiente na modelagem e análise dos processos de negócios. Em [2, 14]

¹do inglês “Flexibility by selection”

²do inglês “Flexibility by adaptation”

³do inglês “Flexibility by design”

⁴do inglês “flexibility by underspecification”

⁵do inglês “flexibility by change”

⁶do inglês “Flexibility by deviation”

e [30] são definidas as *WorkFlow nets*, que são redes de Petri acíclicas que modelam processos de negócios.

Em [31] um modelo estendido de redes de Petri para a modelagem de *workflow* é definido. Tal modelo permite o tratamento de recursos específicos que devem ser utilizados em atividades específicas em tempo real. Em [16], uma extensão das *WorkFlow nets* é apresentada. Este modelo é chamado de *Time WorkFlow net* e associa intervalos de tempo às transições do modelo de rede de Petri correspondente.

A propriedade *soundness* é um critério importante que precisa ser satisfeito quando se trata dos processos de negócios. Na verdade, as “boas” propriedades dos modelos formais bem definidos, tais como as *WorkFlow nets*, podem ser facilmente provadas quando os mesmos respeitam uma rígida estrutura a qual não permite desvios em relação à descrição do processo durante a sua execução em tempo real. No entanto, como as tarefas normalmente são executadas por atores humanos e geralmente são complexas seguindo regras que nem sempre são transformadas em processos computadorizados, assim, os processos de negócios não são então facilmente descritos numa rígida estrutura de modelagem. Logo, as “boas” propriedades envolvidas no modelo não são facilmente comprovadas.

A fim de considerar uma certa flexibilidade na execução dos processos em Sistemas de Gerenciamento de *WorkFlow*, diversos pesquisadores já propuseram diferentes abordagens na literatura científica.

Em [3], especificamente no capítulo 4, Michael Adams descreve a linguagem *Yawl*. O princípio é baseado no *Worklet Service*⁷ que permite a construção de estruturas de subprocessos de tal maneira que elas podem ser adicionadas dinamicamente no processo de negócio durante a sua execução em tempo real. Nessa abordagem, os possíveis desvios do processo são modelados de forma explícita (novos caminhos adicionados), mas a propriedade *soundness* não é garantida.

Uma abordagem para detectar os desvios num processo é apresentado em [32]. Nela, dois modelos coexistem durante a monitoração do processo. Um corresponde ao comportamento normal e pode ser visto como o modelo do processo previsto. O outro é dinamicamente construído através da observação das ações dos atores humanos. Os dois modelos são permanentemente comparados para detectar possíveis desvios. A dificuldade em tal abordagem é lidar simultaneamente com os dois modelos que podem sobrecarregar a atividade de monitoramento reduzindo a performance do sistema.

⁷ *Worklet Service* é um serviço completo, distribuído como parte do ambiente *YAWL*, o qual proporciona flexibilidade dinâmica e suporte ao tratamento de exceção para os processos *Yawl*

Em [13], um tipo de modelo essencialmente baseado em regras é usado para detectar inconsistências (estados inconsistentes com o processo) e aceitar possíveis desvios (transições inconsistentes entre as atividades). A limitação dessa metodologia é relacionada ao modelo do processo que pode ser visto como um simples conjunto de restrições sem uma estrutura real do processo que pode ser analisado a partir do ponto de vista de propriedades básicas.

Cugola et al., em [33] e [34], definem um modelo baseado em lógica temporal e máquina de estados finitos para capturar e tolerar desvios no processo durante a execução. Para os autores, um processo é correto se todas as restrições, dadas pelo conjunto das máquinas de estados, são verificadas. Em particular, dois tipos de transição são criados: as normais e as externas, as quais dependem de requisições do usuário para indicar um comportamento anormal. Além disso, quando a execução do processo é interrompida, o estado do processo é fixado manualmente. O problema com tal abordagem é que o modelo do processo é dado através de uma forma declarativa em vez de um grafo representando todo o processo que poderia ser analisado a partir do ponto de visão da propriedade *soundness*, como é o caso das *WorkFlow nets*. Outro problema é a necessidade de modelar explicitamente os cenários alternativos correspondentes aos comportamentos anormais. A consequência é geralmente o aumento da complexidade do conjunto de restrições e do modelo do processo.

Em [35], [36] e [37], uma abordagem incremental para verificar a conformidade entre o modelo do processo e o *log*⁸ de eventos foi proposta. Inicialmente avalia-se se todas as sequências registradas no *log* são uma sequência de execução possível no que diz respeito ao modelo do processo e, posteriormente avalia-se o grau de precisão entre o modelo do processo e o comportamento esperado. Após a verificação, o analista é auxiliado a localizar as respectivas áreas de não conformidade ou no modelo ou no *log* do processo. Em [38], uma nova forma de medir a conformidade entre o modelo do processo e o *log* foi proposta. Estas quatro abordagens apenas calculam a precisão entre o modelo e a execução do mesmo, permitindo a detecção da inconsistência, após a execução, no *log* ou no modelo do processo.

Em [39], uma nova técnica para medir a precisão do modelo foi proposta. Ela visa complementar a informação de precisão fornecida por outras técnicas, diminuir a complexidade existente na verificação da conformidade entre o modelo e os *logs* e, fornecer informações úteis para uma extensão posterior do processo, ou seja, a fase em que o modelo pode ser estendido para melhor refletir o *log*. Este modelo apenas calcula

⁸Um *log*, em termos gerais, é um registro de acontecimentos ou de atividades. Em se tratando de computadores, é um registro gerado por um serviço ou aplicativo específico (às vezes pelo próprio sistema operacional).

a precisão do modelo guiando-o para uma futura alteração, tornando-o mais preciso em relação ao *log*, não tratando em tempo real as inconsistências geradas pelos atores humanos.

Uma coleção de algoritmos de mineração de processos é descrita em [40], visando a verificação da conformidade e da análise de desempenho do processo. Foram consideradas quatro dimensões de qualidade para comparar o modelo e o *log*: adequação, simplicidade, precisão e generalização. Muitas das métricas apresentadas neste artigo foram implementadas no *ProM*, uma plataforma *open source* de mineração de processo que oferece uma grande variedade de técnicas para a verificação de conformidade e aprimoramento do modelo. Esta abordagem apenas mede a precisão do modelo do processo em relação ao *log*.

Em [41], uma abordagem para lidar com os desvios dos agentes durante a execução do processo foi proposta. Esta abordagem monitora a execução do processo obtendo os rastros de execução das ações realizadas pelo agente e comparando-os com as descrições das ações permitidas fornecidas pelo modelo do processo. A partir desta comparação, se um desvio for identificado, o agente será alertado e ele poderá corrigi-lo ou ignorá-lo. Um problema existente é o tempo de processamento da comparação, reduzindo bastante a performance do sistema.

Zazworka et al. definiram uma abordagem para detectar as não conformidades nos processos de treinamento dos desenvolvedores num ambiente empresarial ou acadêmico em [12]. Inicialmente as regras de conformidade são definidas e, posteriormente, os dados são capturados de forma não intrusiva durante a execução do processo. Uma análise é realizada a partir das regras e dos dados e o resultado é informado ao gerente de projeto, permitindo-o um possível redirecionamento do processo. Uma limitação dessa abordagem é a falta de exatidão na análise e na coleta dos dados, principalmente quando os dados são coletados manualmente.

Nas abordagens propostas em [3, 12, 13, 32, 33, 34, 35, 39, 40, 41], no que se diz respeito a flexibilidade nos processos de negócio, todas apenas detectam a inconsistência gerada e não a trata imediatamente após a ocorrência da mesma.

Uma alternativa muito promissora para lidar com a flexibilidade nos processos de negócios são as abordagens baseadas em raciocínio incerto. Os conceitos de conjuntos nebulosos [17], da lógica *fuzzy* [21] e da lógica possibilística [22], já foram contemplados em numerosas áreas da computação e da engenharia quando informações imprecisas devem ser levadas em conta.

Zadeh apresenta algumas ideias básicas e explicações representativas subjacentes à lógica *fuzzy*, em [21], e sobre os conjuntos nebulosos, em [17]. Dubois et al., em [22], expõe a lógica possibilística cobrindo os aspectos formais e algumas aplicações da lógica em sistemas onde a incerteza pode ser considerada.

Cardoso et al. em [6, 9, 20] e Murata et al. em [19, 42] utilizaram as redes de Petri juntamente com a lógica possibilística e conjuntos nebulosos para incluir uma noção de incerteza nos processos da manufatura permitindo uma maior flexibilidade e modelos mais compactos.

Em [20] e [6], a marcação incerta, definida por distribuições de possibilidades, e a noção de duração *fuzzy*, caracterizada por dados *fuzzy*, são definidas nas “redes de Petri a Objetos” para “relaxar” as relações de sincronização e/ou descrever um conjunto de alternativas possíveis. A marcação incerta é usada para supervisionar o ambiente físico e descrever a informação incompleta sobre o estado corrente do sistema. A duração *fuzzy* é associada a cada transição permitindo supervisionar a execução do tempo programado.

Em [9], a “rede de Petri possibilística”, o qual combina a lógica possibilística com a “rede de Petri a Objetos”, é apresentada. Nesse modelo, os estados são indicados por distribuições de possibilidade e a evolução do estado do processo é visto como uma atualização dessa distribuição de possibilidade por meio dos “tipos de disparo” da transição. A principal característica desse modelo é permitir raciocinar sobre os aspectos de incerteza e trocas num sistema dinâmico a eventos discretos.

Em [19] e [42], a noção explícita de tempo são incluídas nas redes de Petri. A principal característica é a introdução de quatro funções teóricas do conjunto *fuzzy* relacionadas ao tempo. Elas são: *fuzzy timestamp*, *fuzzy enabling time*, *fuzzy occurrence time* e *fuzzy delay*, todas capturando a incerteza temporal.

Jeske et al., em [43], propuseram a “rede de Petri p-temporal” com recursos híbridos *fuzzy* como solução para o problema de alocação de recursos humanos em Sistemas de Gerenciamento de *Workflow*. Para expressar de forma mais realista os mecanismos de alocação de recursos onde o comportamento humano é considerado, conjuntos *fuzzy*, delimitados por distribuições de possibilidade na forma triangular, foram associados com as marcações dos lugares que representam a disponibilidade humana. Novas regras de disparo e evolução das marcações *fuzzy* do novo modelo foram então definidas. O problema específico de desvio entre o modelo do processo e a execução do mesmo não foi contemplado neste trabalho.

Cimpan e Oquendo, em [44], propuseram um modelo do processo dado por meio de conjuntos *fuzzy* e distribuições de possibilidade. A vantagem do raciocínio *fuzzy* ou

possibilístico é a habilidade de representar naturalmente a informação incerta ou imprecisa que existe quando as atividades são executadas por humanos. Uma limitação desta abordagem é a falta de uma representação formal e gráfica que pode ser analisada do ponto de vista das boas propriedades que um processo de negócio deve apresentar.

2.1 Considerações Finais do Capítulo

Este capítulo apresentou as principais contribuições relacionadas ao tratamento dos desvios e inconsistências nos processos de negócios, informando as limitações encontradas em cada proposta.

Diversas contribuições sobre flexibilidade nos processos de negócios foram propostas com a finalidade de tornar o modelo do processo condizente com a sua execução. Na maioria delas, os desvios/inconsistências geradas a partir da execução dos modelos dos processos são apenas detectadas, ou seja, a recuperação do estado do processo é realizado manualmente pelos agentes responsáveis. Além disso, algumas propostas focaram em medir a conformidade do modelo do processo em relação ao seu log de eventos ou aos seus rastros de execução das ações executadas.

Na maioria, as “boas” propriedades, tais como a propriedade *sound*, não são garantidas. Além disso, outras modificam o modelo em tempo de execução, reduzindo assim a performance do mesmo.

Contudo, no modelo proposto neste trabalho, os desvios/inconsistências ocasionadas a partir da execução do processo por atores humanos serão tratadas, em tempo de execução, sem alterar o modelo definido. Ademais, as boas propriedades existentes serão garantidas.

Capítulo 3

Fundamentos Teóricos

Este capítulo apresenta os fundamentos teóricos necessários para o entendimento deste trabalho.

3.1 Redes de Petri Clássica

A teoria inicial das redes de Petri foi apresentada em 1962 por Carl Adam Petri em [45]. Uma rede de Petri [15, 46] é uma representação gráfica e um modelo formal abstrato que pode ser utilizada para a modelagem de diversos tipos de sistemas, em particular os sistemas a eventos discretos. O fato de ser formal permite, entre outras características, a verificação formal de certas propriedades.

A definição de uma “rede de Petri clássica” é a seguinte [47]:

Definição 3.1. (“Rede de Petri Clássica”) Uma “rede de Petri clássica” é uma quádrupla $(P, T, \text{Pré}, \text{Pós})$ onde:

- P é um conjunto finito de lugares;
- T é um conjunto finito de transições;
- Pré é uma relação que define os arcos que ligam os lugares às transições;
- Pós é uma relação que define os arcos que ligam as transições aos lugares.

Um lugar p é chamado de lugar de entrada de uma transição t se, e somente se, existe um arco direcionado de p para t . Por exemplo, considerando a rede de Petri da Figura 3.1(a), os lugares P_1 e P_2 são lugares de entrada da transição t_1 .

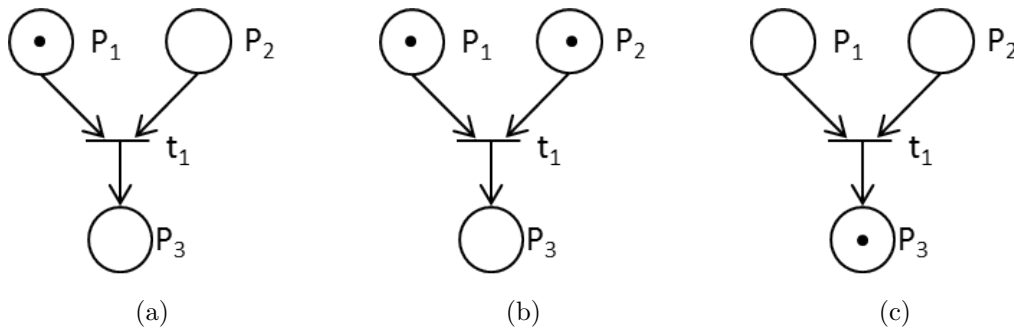


FIGURA 3.1: Exemplos de sensibilização e disparo de transição em uma rede de Petri.

Um lugar p é chamado de lugar de saída de uma transição t se, e somente se, existe um arco direcionado de t para p . Por exemplo, considerando a rede de Petri da Figura 3.1(a), o lugar P_3 é um lugar de saída da transição t_1 .

Um lugar p , quando se considera as redes de Petri clássica marcadas, contém, em um dado momento, zero ou mais fichas (*tokens*) representadas por pontos pretos (\bullet). Por exemplo, considerando a rede de Petri da Figura 3.1(a), o lugar P_1 contém uma ficha e os lugares P_2 e P_3 contêm zero fichas.

A marcação de uma “rede de Petri clássica” refere-se à distribuição de fichas nos lugares, sendo que o número de fichas pode mudar durante a execução da rede.

As transições são componentes ativos em uma “rede de Petri clássica”: elas mudam a marcação da rede de acordo com as seguintes regras de disparo:

- uma transição t é dita sensibilizada se, e somente se, cada lugar de entrada p de t contém pelo menos uma ficha. Por exemplo, a transição t_1 da rede de Petri da Figura 3.1(a) não está sensibilizada. Já a transição t_1 da rede de Petri da Figura 3.1(b) está sensibilizada, pois há uma ficha em cada lugar de entrada desta transição;
- uma transição sensibilizada pode disparar. Se a transição t disparar, então t consome uma ficha de cada lugar de entrada p de t e produz uma ficha em cada lugar de saída p de t . As redes de Petri das Figuras 3.1(b) e 3.1(c) mostram um exemplo de disparo de transição, isto é, a marcação inicial é representada pela Figura 3.1(b) e a marcação final, obtida após o disparo da transição t_1 , é representada pela Figura 3.1(c). Neste caso, t_1 consome uma ficha de cada lugar de entrada (P_1 e P_2) e produz uma ficha em cada lugar de saída (P_3).

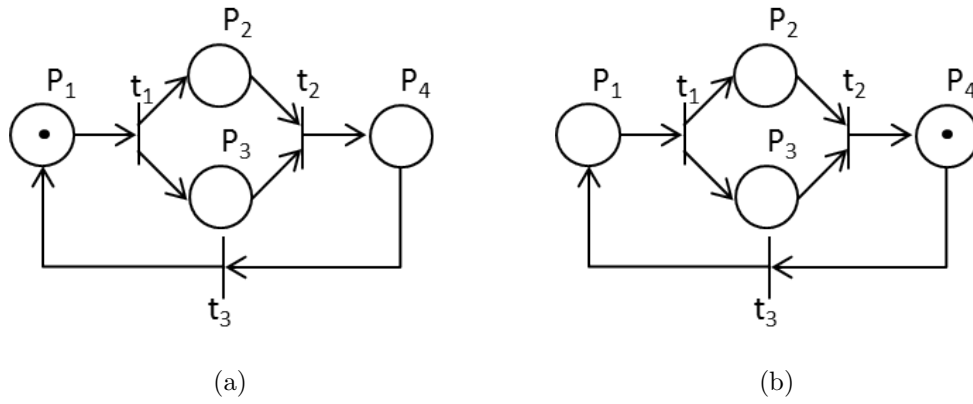


FIGURA 3.2: Sequência de disparo de transições.

Uma “rede de Petri clássica” pode ser representada algebricamente por uma matriz de incidência, a qual representa a relação entre os lugares e as transições, com dimensão $n \times m$: o número de linhas é igual ao número de lugares e o número de colunas é igual ao número de transições. Esta matriz é formada pela subtração da matriz de incidência posterior $Pós$ com a matriz de incidência anterior $Pré$ ($I = Pós - Pré$). A notação matricial da rede de Petri da Figura 3.2(a) é dada por:

$$Pré = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Pós = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad \text{e} \quad I = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

A matriz de incidência I fornece o balanço das fichas na rede quando ocorre o disparo das transições. Cada coluna corresponde à modificação da marcação através do disparo da transição associada. Por exemplo, a primeira coluna da matriz de incidência I indica que o disparo da transição t_1 consiste da remoção de uma ficha do lugar P_1 e da adição de uma ficha em cada um dos lugares P_2 e P_3 .

Uma sequência de disparo S pode ser representada por um vetor chamado *vetor característico* da sequência S . Sua dimensão é igual ao número de transições da rede de Petri e cada componente pertencente à S representa o número de ocorrências da transição t numa sequência de disparo S . Considerando a Figura 3.2(a), dois possíveis vetores característicos das sequências $s_1 = t_1 t_2 t_3$ e $s_2 = t_1 t_2$ são representados matricialmente por:

$$s_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{e} \quad s_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Se o disparo de uma sequência S é tal que a partir de uma marcação inicial M_i obtemos uma marcação final M_f , então a **Equação Fundamental** da “rede de Petri clássica” é dada por:

$$M_f = M_i + I \cdot S \quad (3.1)$$

o que significa que se pode obter uma determinada marcação final¹ M_f , a partir de uma marcação inicial² M_i , através da adição da marcação inicial ao produto da matriz de incidência pelo vetor característico.

A equação fundamental descreve o comportamento da “rede de Petri clássica” possibilitando a sua análise estrutural e comportamental. Para exemplificar o funcionamento de tal equação, a Figura 3.2(a) e a sequência de disparo s_2 serão consideradas. A marcação final M_f obtida é apresentada abaixo (a qual pode ser observada na Figura 3.2(b)):

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

O vetor característico S nem sempre corresponde a uma sequência de disparo válida e nem informa a ordem da execução das transições, entretanto, constitui uma condição necessária para o problema de alcançabilidade [46].

3.2 Redes de Petri a Objetos

Antes de definir as redes de Petri a Objetos, é importante considerar a noção de “objeto”. O conceito de objeto, cada vez mais utilizado em Engenharia de *Software*, consiste em estruturar uma aplicação em torno das entidades envolvidas, encapsulando, ao mesmo tempo, estruturas de dados, sob a forma de uma lista de atributos e métodos de transformação destes dados. Esta abordagem opõe-se ao procedimento funcional segundo o qual se definem de maneira separada as estruturas de dados e as funções que o sistema deve executar [46].

Uma classe de objetos é definida por um conjunto de atributos (também chamados propriedades) e um conjunto de operações ou métodos que permitem manipular os valores dos atributos. Pode-se definir uma classe a partir de outra por herança. A nova

¹A marcação final representa o estado final do sistema correspondente

²A marcação inicial representa o estado inicial do sistema correspondente

classe (chamada subclasse) herda as definições de atributos da classe anteriormente definida. Estas definições podem ser completadas por atributos e operações específicas.

As classes são apenas definições. Um objeto particular é uma instância de classe de objeto. Pode-se atribuir valores aos atributos destes objetos e executar suas operações. O encapsulamento confere aos objetos uma certa autonomia e uma certa remanescência. Um objeto nasce (instanciação) e desaparece (destruição) dinamicamente durante a execução do programa, mas tais eventos devem ser raros em relação ao número de eventos correspondendo à execução de operações sobre os atributos.

As redes de Petri a Objetos definidas por Sibertin-Blanc [48] baseiam-se na integração das “redes de Petri Predicado/Transição” e do conceito de paradigma “orientado a objetos”. Neste tipo de rede, as fichas são consideradas como n-uplas de instâncias de classes de objetos e transportam estruturas de dados definidas como conjuntos de atributos de classes específicas. Nas transições, por sua vez, são associadas pré-condições (filtros) e ações, que respectivamente, atuam sobre os atributos das fichas e modificam seus valores [46].

Em uma “rede de Petri a Objetos”, os lugares estão associados às classes de objetos, as transições são associadas às operações que atuam sobre os atributos dos objetos situados nos lugares de entrada, e os objetos correspondem às fichas da rede. Dessa forma, a operação associada a uma determinada transição t só poderá ser executada por um objeto se este estiver num lugar de entrada de t [46]. As “redes de Petri a Objetos” podem ser definidas formalmente como:

Definição 3.2. (“Rede de Petri a Objetos”) Uma “rede de Petri a Objetos” [46] marcada pode ser definida pela 9-upla:

$$R = \langle P, T, C_{class}, V, Pré, Pós, A_{tc}, A_{ta}, M_0 \rangle$$

onde:

- C_{class} representa um conjunto finito de classes de objetos; para cada classe um conjunto de atributos é definido e eventualmente organizado em uma hierarquia;
- P é um conjunto finito de lugares, cujos tipos são dados por C_{class} ;
- T é um conjunto finito de transições;
- V é um conjunto de variáveis formais, cujos tipos são dados por C_{class} ;

- $Pré$ é a função “*lugar precedente*” que, a cada arco de entrada de uma transição, faz corresponder uma soma formal de elementos de V ;
- $Pós$ é a função “*lugar seguinte*” que, a cada arco de saída de uma transição, faz corresponder uma soma formal de elementos de V ;
- A_{tc} é uma aplicação que, a cada transição, associa uma condição que envolve o conjunto de atributos e métodos dos objetos por meio das variáveis formais V associados aos arcos de entrada;
- A_{ta} é uma aplicação que, a cada transição, associa uma ação que envolve os atributos ou métodos das variáveis formais associados aos arcos de entrada permitindo modificar seus atributos específicos por meio da invocação de seus métodos;
- M_0 é a marcação inicial que associa, a cada lugar, uma soma formal dos objetos (n-uplas de instâncias de classes que pertencem a C_{lass}).

Um exemplo de “rede de Petri a Objetos” é representada na Figura 3.3. Para esta rede são definidos:

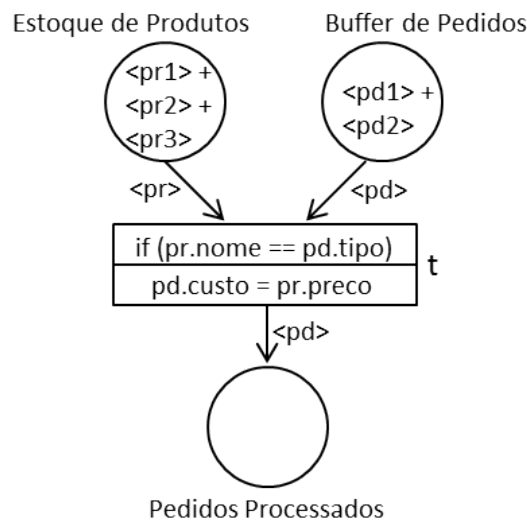


FIGURA 3.3: Exemplo de uma “rede de Petri a Objetos”.

- o conjunto de classes $C_{lass} = \{Produto, Pedido\}$ com:

$$\begin{aligned}
 \text{Produto} &= \left\{ \begin{array}{l} \text{nome} : \text{identificador}; \\ \text{codigo} : \text{integer}; \\ \text{preco} : \text{float}; \end{array} \right. & \text{Pedido} &= \left\{ \begin{array}{l} \text{codigo} : \text{integer}; \\ \text{custo} : \text{float}; \\ \text{tipo} : \text{identificador}; \end{array} \right.
 \end{aligned}$$

- as variáveis pr , de tipo *Produto* e pd , de tipo *Pedido*: $tipo(pr) = Produto$ e $tipo(pd) = Pedido$;
- os lugares possuem os tipos: $tipo(Estoque\ de\ Produtos) = Produto$, $tipo(Bufer\ de\ Pedidos) = Pedido$ e $tipo(Pedidos\ Processados) = Pedido$;
- a marcação inicial M_0 é dada pelos objetos que se encontram nos lugares:

$$M_0 = \begin{bmatrix} < pr1 > + < pr2 > + < pr3 > \\ < pd1 > + < pd2 > \\ 0 \end{bmatrix}$$

em que os objetos $< pr1 >$, $< pr2 >$, $< pr3 >$ são da classe *Produto*, e os objetos $< pd1 >$ e $< pd2 >$ são da classe *Pedido*.

Nos lugares “*Estoque de Produto*” e “*Buffer de Pedidos*”, os objetos (fichas) possuem valores de atributos. Por exemplo, para o objeto $< pr1 >$, os valores de atributos podem ser dados por:

$$\text{Produto } pr1 = \begin{cases} nome : home\ theater; \\ codigo : 123440; \\ preco : 278,50; \end{cases}$$

E para o objeto $< pd2 >$, valores dos atributos podem ser dados por:

$$\text{Pedido } pd2 = \begin{cases} codigo : 123440; \\ custo : 00,00; \\ tipo : home\ theater; \end{cases}$$

A definição detalhada que fixa as regras de sensibilização e disparo das transições das redes de Petri a Objetos encontram-se em [48].

As variáveis do arco de entrada da transição podem ser substituídas por variáveis (objetos) de mesmo tipo, que se encontram nos lugares de entrada da transição. Entretanto, é necessário que seja satisfeita a pré-condição da transição. Se esta pré-condição é atendida, o disparo da transição remove os objetos dos lugares de entrada da transição, altera os valores de atributos de acordo com a ação existente na transição e, em seguida, produz um novo objeto com valores atualizados nos lugares de saídas correspondentes.

Pode-se notar que, na Figura 3.3, a transição t pode ser sensibilizada pela marcação inicial. Os valores dos atributos da variável pr associada ao arco que liga o lugar

“*Estoque de Produtos*” à transição t podem ser substituídos pelos valores de atributos de um dos objetos que se encontram em “*Estoque de Produtos*”. Da mesma forma, os valores dos atributos da variável pd associada ao arco que liga o lugar “*Buffer de Pedidos*” à transição t podem ser substituídos pelos valores de atributos de um dos objetos que se encontram em “*Buffer de Pedidos*”.

Inicialmente, a pré-condição associada à transição t deve ser satisfeita para que a transição t seja disparada. A pré-condição “if ($pr : nome == pd : tipo$)” verifica se os atributos “*nome*”, da variável pr , e “*tipo*”, da variável pd , possuem os mesmos valores. Se a condição é satisfeita, a transição t é disparada. A ação associada à transição é então executada registrando o valor do atributo “*preco*”, da variável pr , no atributo “*custo*”, da variável pd . Após o disparo da transição t , um novo objeto, com o valor do atributo “*custo*” modificado, é produzido no lugar “*Pedidos Processados*”.

Na “rede de Petri a Objetos” ilustrada pela Figura 3.3, o par de objetos ($pr1, pd2$) verifica a condição associada à transição t , logo a transição é disparada, produzindo o objeto $pd2$ no lugar “*Pedidos Processados*”. A Figura 3.4 ilustra a “rede de Petri a Objetos” com a nova marcação.

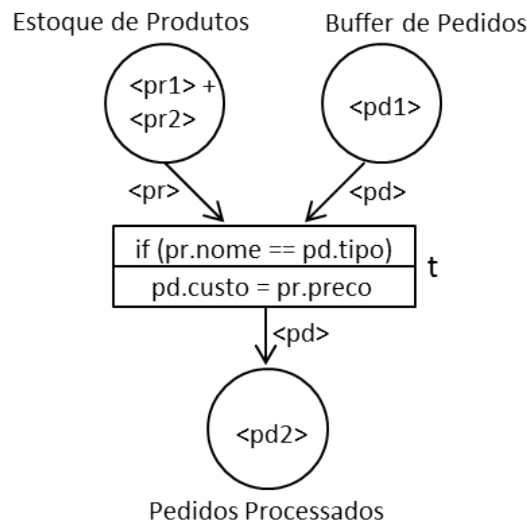


FIGURA 3.4: Exemplo de disparo da transição t .

Para os demais objetos da rede, os disparos podem ser realizados na mesma forma.

3.3 Redes de Petri Possibilísticas

A “rede de Petri possibilística” combina a teoria da possibilidade [22] com a teoria das redes de Petri a Objeto [48]. Ela é um modelo onde um lugar, uma transição e

uma sequência de disparo denotam, respectivamente, um possível estado parcial, uma possível troca de estado e um possível comportamento [20]. A principal vantagem é a capacidade de atualizar o estado do sistema com informações incompletas sem originar estados inconsistentes. As “redes de Petri possibilísticas” podem ser definidas formalmente a partir das redes de Petri a Objeto (Definição 3.2).

Por elas serem derivadas das “redes de Petri a Objeto” [49], as fichas são instâncias de classes ou subclasses de objetos e seus atributos podem estar envolvidos nos predicados associados às transições atuando como condições extras no disparo das mesmas. Elas também podem ser modificadas pela execução de uma ação quando a transição correspondente é disparada.

A fim de representar a incerteza, a “rede de Petri possibilística” associa uma distribuição de possibilidade $\pi_o(p)$ [22] ao estado ou lugar onde está o objeto o (p corresponde à um lugar da rede). A partir disso, as seguintes declarações podem ser enunciadas:

- $\pi_o(p) = 1$ representa o fato de que existe possibilidade de o estar no lugar p ;
- $\pi_o(p) = 0$ expressa a certeza de que o não está presente no lugar p ;
- num dado momento, $\pi_o(p) = 1$ e $\pi_o(p') = 1$ para $p \neq p'$ pode acontecer.

Considerando as declarações acima, dizer que é “completamente possível” (possibilidade igual a 1) que o objeto o está no lugar p é muito menos relevante do que dizer que é “completamente certo” que o objeto o está no lugar p . De fato, a primeira declaração não proíbe que o objeto o esteja em outro lugar p' para, ou seja, não inibe que o estado de o seja incerto. Logo, uma distribuição de possibilidade permite modelar:

- *uma marcação precisa*: onde cada ficha está em apenas um lugar (estado bem conhecido);
- *uma marcação imprecisa*: onde se tem uma distribuição de possibilidade da existência da ficha sobre um conjunto de lugares. Não se pode afirmar que uma ficha está num dado lugar, mas apenas que está em um lugar dentre um conjunto de lugares.

Formalmente, uma marcação numa “rede de Petri possibilística” é, então, um mapeamento:

$$M : O \times P \longrightarrow \{0, 1\} \quad (3.2)$$

onde O é um conjunto de objetos e P um conjunto de lugares. Se $M(o, p) = 1$, existe uma possibilidade do objeto o estar no lugar p . Caso contrário, $M(o, p) = 0$ e não existe a possibilidade do objeto o estar em p .

A marcação M da rede permite representar, formalmente:

- *uma marcação precisa*: $M(o, p_i) = 1$ e $\forall p_j \neq p_i, M(o, p_j) = 0$;
- *uma marcação imprecisa*: $M(o, p_i) = 1$ e $\exists p_j \neq p_i, M(o, p_j) = 1$.

O disparo de uma transição t qualquer é decomposta em dois passos:

- *início do disparo*: as fichas são colocadas nos lugares de saída de t , mas não são removidas de seus lugares de entrada;
- *término do disparo*: pode ser o cancelamento do disparo (as fichas são removidas dos lugares de saída de t), ou o arquivamento do disparo (as fichas são removidas dos lugares de entrada de t).

Numa rede de Petri com marcação imprecisa, as transições podem ser disparadas de duas maneiras diferentes. São elas:

- *disparo certo*: corresponde ao disparo de uma transição como na “rede de Petri clássica”, o início e o arquivamento imediato do mesmo. Neste caso, o posicionamento nos lugares de todos os objetos envolvidos no disparo deve ser necessariamente certo. A ação associada à transição é executada;
- *disparo incerto* (ou *pseudo-disparo*): não corresponde a uma evolução normal do sistema. Considera-se apenas o início do disparo onde algumas deduções sobre o estado do sistema são realizadas. As ações associadas com a transição pseudo-disparada não são realizadas uma vez que o comando atual poderá apenas ser concretizado quando o conhecimento for certo. A incerteza sobre o posicionamento nos lugares desses objetos irá aumentar até que a ocorrência de um evento permita deduzir o posicionamento dos objetos nos lugares com certeza possibilitando retornar ao conhecimento certo do sistema.

Após o disparo incerto, a distribuição de possibilidade de cada objeto presente nos lugares de entrada e de saída da transição é igual à 1. Este disparo permite evitar a ocorrência de contradições no caso de um comportamento anormal do sistema. Assim

que um incidente ocorre, o conhecimento incerto é utilizado a fim de levar em conta um conjunto maior de possíveis eventos.

Neste tipo de disparo uma restrição é imposta: a distribuição de possibilidade dos objetos pertencentes aos lugares de saída da transição a ser pseudo-disparada deve ser igual à 0. Caso essa restrição seja violada, o algoritmo de defuzzificação, detalhado na Seção 3.3.1, poderá vir a ser impossível, pois, possivelmente, haverá inúmeras sequências de disparos que poderão alcançar a nova marcação a partir da última marcação certa.

A interpretação de uma “rede de Petri possibilística” é definida anexando a cada transição uma função de autorização η_{x_1, \dots, x_n} , a qual representa uma condição extra:

$$\eta_{x_1, \dots, x_n} : T \longrightarrow \{Falsa, Incerta, Verdadeira\} \quad (3.3)$$

onde x_1, \dots, x_n são as variáveis associadas aos arcos de entrada da transição t (quando se considera uma “rede de Petri a Objetos”).

Considerando uma transição t qualquer e uma possível substituição o_1, \dots, o_n para as possíveis variáveis x_1, \dots, x_n associadas aos arcos de entrada desta transição, diversas situações podem ser consideradas:

- t não está habilitada pela marcação mas sua interpretação é verdadeira, essa situação é proibida e um alarme é ativado;
- t está habilitada por uma marcação precisa e uma interpretação verdadeira, então um disparo como na “rede de Petri clássica” (com certeza) ocorrerá;
- t está habilitada por uma marcação precisa e uma interpretação incerta, então a transição t é pseudo-disparada e a imprecisão aumentará;
- t está habilitada por uma marcação imprecisa, se a interpretação é incerta então t é pseudo-disparada;
- t está habilitada por uma marcação imprecisa e uma interpretação verdadeira: o algoritmo de defuzzificação, detalhado na Seção 3.3.1, é chamado e uma nova computação da distribuição de possibilidade dos objetos envolvidos na marcação imprecisa é realizado a fim de voltar a uma marcação precisa.

Um exemplo de “rede de Petri possibilística” é representada na Figura 3.5. Para esta rede são definidos:

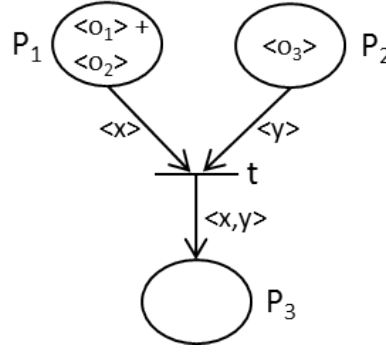


FIGURA 3.5: Exemplo de “rede de Petri possibilística”.

- o conjunto de classes $C_{lass} = \{Classe_1, Classe_2, (Classe_1, Classe_2)\}$ com:

$$Classe_1 = \left\{ \begin{array}{l} data : date; \end{array} \right.$$

Nenhum atributo ou função é definido para a classe $Classe_2$, pois, para o entendimento do exemplo, não é necessário tal detalhamento.

- as variáveis x , de tipo $Classe_1$ e y de tipo $Classe_2$: $tipo(x) = Classe_1$ e $tipo(y) = Classe_2$;
- os lugares possuem os tipos: $tipo(P_1) = Classe_1$, $tipo(P_2) = Classe_2$ e $tipo(P_3) = (Classe_1, Classe_2)$;
- a marcação inicial M_0 é dada pelos objetos que se encontram nos lugares:

$$M_0 = \begin{bmatrix} < o_1 > + < o_2 > \\ < o_3 > \\ 0 \end{bmatrix}$$

em que os objetos $< o_1 >$ e $< o_2 >$ são da classe $Classe_1$, e o objeto $< o_3 >$ é da classe $Classe_2$;

- a função de autorização é dada por:

$$\eta_{x,y} = \forall_y \begin{cases} incerto & se \quad (\tau < x.date) \wedge (signal(x)) \\ verdadeiro & se \quad (\tau \geq x.date) \wedge (signal(x)) \\ falso & caso \quad contrario \end{cases} \quad (3.4)$$

onde, x e y são as variáveis associadas ao arco de entrada da transição t e, τ o tempo corrente.

A função de interpretação impõe à transição t restrições para o disparo da mesma. O disparo como na “rede de Petri clássica” acontece quando a chegada da mensagem

“ $signal(x) = verdade$ ” ocorre depois do tempo determinado, representado pelo atributo “ $data$ ”, sinalizando que o objeto x foi envolvido em um evento associado à transição t . O disparo incerto ocorre quando a mensagem chega antes do tempo determinado, um comportamento que é possível, mas não normal. Este disparo permite deduzir que a mensagem é errônea ou que o estado do processo atual não condiz com a representação do mesmo no modelo (a rede de Petri marcada).

Considerando a marcação inicial da Figura 3.5 com duas possíveis substituições para as variáveis associadas aos arcos de entrada da transição t ($S_1 = (o_1, o_3)$ e $S_2 = (o_2, o_3)$) e assumindo que $o_1.date = 20$, $o_2.date = 40$, no tempo $\tau_1 = 25$, $signal(o_2) = verdade$ e no tempo $\tau_2 = 35$, $signal(o_1) = verdade$. Seguem-se as considerações em função desse tempo:

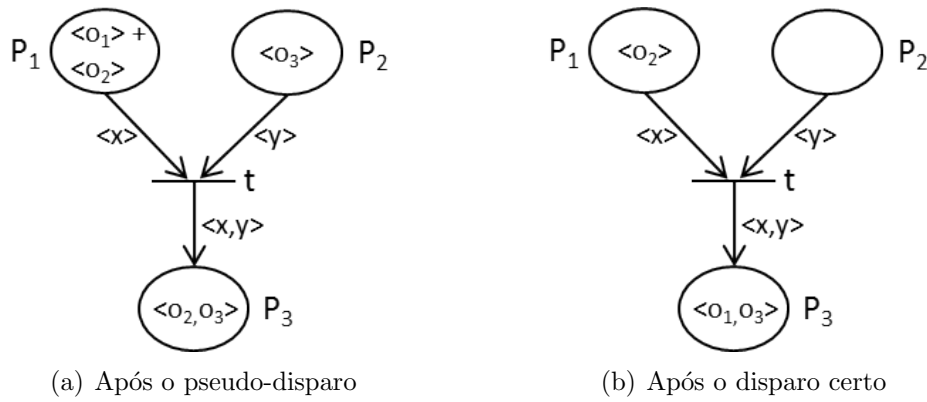


FIGURA 3.6: Marcação.

- no tempo $\tau = 25$, $signal(o_2) = verdade$ é recebido mas não corresponde ao comportamento preciso ($o_2.date > 25$); $\eta_{o_2 o_3}(t) = incerto$ (Equação 3.4) e t é pseudo-disparada com a substituição S_2 (Figura 3.6(a));
- após o tempo $\tau = 25$ a marcação é imprecisa e cobre duas alternativas:
 1. o evento correspondente ao disparo de t para a tupla $\langle o_2, o_3 \rangle$ realmente ocorreu; a informação dada por $signal(o_2)$ foi correta;
 2. o evento correspondente ao disparo de t para a tupla $\langle o_2, o_3 \rangle$ não ocorreu; essa transição ainda pode ser disparada, com $\langle o_1, o_3 \rangle$ ou com $\langle o_2, o_3 \rangle$.
- no tempo $\eta = 35$, o recebimento de $signal(o_1) = verdade$ corresponde ao comportamento preciso ($o_1.date \leq 35$) e $\eta_{o_1 o_3} = verdadeira$. t é então habilitada por uma marcação imprecisa com uma interpretação verdadeira. Consequentemente, o algoritmo de defuzzificação, detalhado na Seção 3.3.1, é chamado para retornar a uma marcação precisa, cancelando o pseudo-disparo de t . Como a marcação

agora é precisa, t é disparado com certeza utilizando a substituição S_1 (Figura 3.6(b)).

A figura (3.7) descreve a distribuição das possibilidades das instâncias o_1, o_2 e o_3 em função do tempo (as linhas mais espessas representam uma possibilidade igual a 1 e as linhas mais finas uma possibilidade igual a 0).

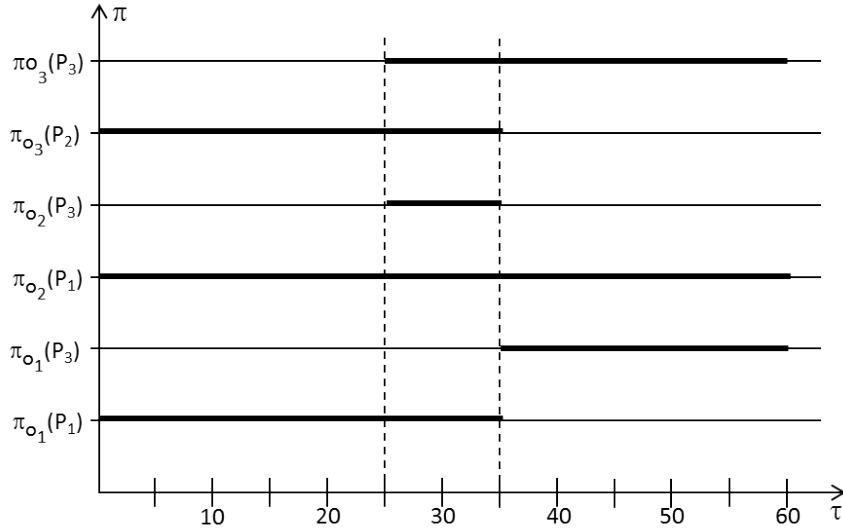


FIGURA 3.7: Distribuição das possibilidades das localizações de o_1, o_2 e o_3 .

3.3.1 Algoritmo de Defuzzificação

O restabelecimento do estado certo do sistema ocorrerá quando a função de autorização de uma transição t ($\eta(t)$) for verdadeira e o posicionamento dos objetos nos lugares relacionados à ela for incerto. Entretanto, este algoritmo recupera apenas o estado certo dos objetos envolvidos neste evento, ou seja, caso existam outros pseudo-disparos que não se relacionem direta ou indiretamente a este evento os mesmos não serão recuperados, continuando assim numa situação de imprecisão do conhecimento.

Como definido anteriormente na Seção 3.3, o disparo como na “rede de Petri clássica” ocorre quando a localização dos objetos relacionados ao disparo é certa, ou seja, a distribuição de possibilidade de um objeto o qualquer (π_o) é igual a 1 em apenas um lugar. Consequentemente, é necessário ter um procedimento que realize uma nova computação das distribuições de possibilidade que alcance esse objetivo, levando em conta a nova informação recebida, ou seja, o fato que $\eta(t)$ é verdadeira.

O pseudo-disparo de uma transição t , como mencionado anteriormente, pode ser considerado apenas como o início de um disparo normal. De fato, as fichas são adicionadas

nos lugares de saída, mas não são removidas dos seus lugares de entrada. Isso ocorre pela falta de certeza se a transição t foi ou não realmente disparada.

Ao realizar o novo cálculo das distribuições de possibilidade, deve-se decidir, para cada transição que foi pseudo-disparada, se as mesmas foram ou não efetivamente disparadas. No primeiro caso considera-se que o disparo foi arquivado (t realmente foi disparada) e, no segundo, que o disparo foi cancelado (t não foi disparada). Em outras palavras:

- pseudo-disparo arquivado: a transição foi disparada, pois o evento correspondente realmente ocorreu. As distribuições de possibilidade dos objetos envolvidos no disparo são redefinidas para 0 (zero) nos lugares de entrada da transição;
- pseudo-disparo cancelado: a transição não foi disparada pois o evento correspondente não ocorreu. As distribuições de possibilidade dos objetos envolvidos no disparo são redefinidas para 0 (zero) nos lugares de saída da transição.

Para que o algoritmo de defuzzificação funcione, nenhum disparo incerto poderá ser realizado caso a distribuição de possibilidade dos objetos pertencentes aos lugares de saída da transição seja igual à 1 e uma transição não pode ser pseudo-disparada por mais de uma vez durante um intervalo de tempo. Portanto, o algoritmo de defuzzificação, apresentado em [50], é descrito abaixo (considere LL uma lista de lugares):

Método de Defuzzificação

Entrada: Rede de Petri com posicionamento incerto dos objetos nos lugares, transição t

Saída: Rede de Petri com posicionamento certo dos objetos nos lugares

Se t foi pseudo-disparada anteriormente ($\eta(t) = incerta$)

então $LL \leftarrow Pré(t) \cup Pós(t)$

cancelar o disparo de t

senão $LL \leftarrow Pré(t)$

Enquanto $LL \neq \{\}$

faça $p \leftarrow LL[0]$

Enquanto $\exists t_i \in Pré(p) \mid \eta(t_i) = incerta$

faça $LL \leftarrow LL \cup Pré(t_i)$

arquivar o disparo de t_i

Enquanto $\exists t_j \in Pós(p) \mid \eta(t_j) = incerta$

faça $LL \leftarrow LL \cup Pós(t_j)$

cancelar o disparo de t_j

Disparar como na “rede de Petri clássica” a transição t

Este algoritmo sempre para, pois o número de lugares e transições é finito e por assumir que não existe um ciclo de possíveis lugares para um objeto, pois nenhum pseudo-disparo é realizado quando a distribuição de possibilidade dos objetos é igual à 1 nos lugares de saída da transição.

Para ilustrar o funcionamento do algoritmo, a “rede de Petri possibilística” apresentada na Figura 3.8 será utilizada. Neste exemplo, as transições t_1 , t_3 , t_4 , t_5 e t_8 foram pseudo-disparadas e, após tais disparos, a interpretação da transição t_3 ($\eta(t_3)$) tornou-se verdadeira.

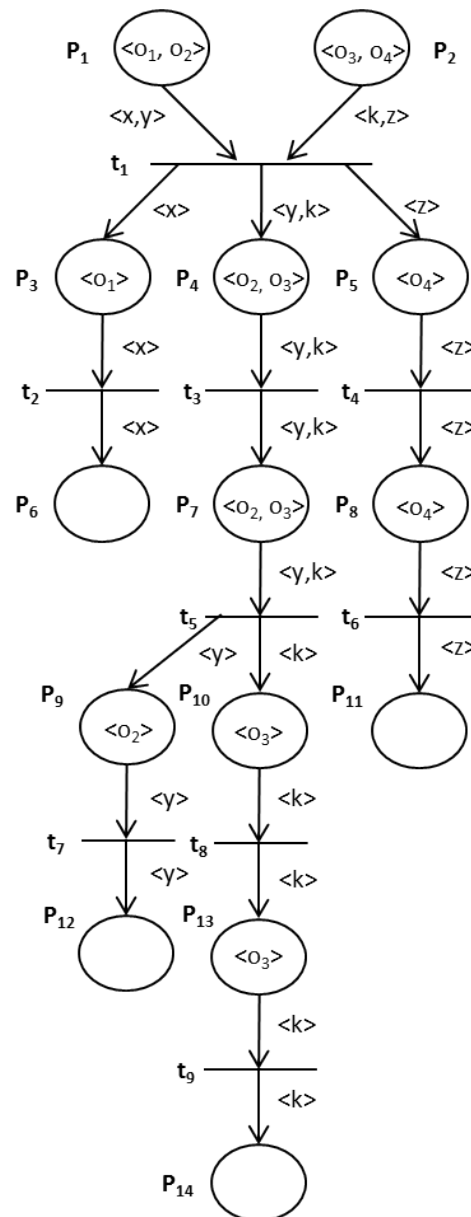


FIGURA 3.8: Exemplo de uma “rede de Petri possibilística” para uso do algoritmo de defuzzificação.

Aplicando o algoritmo de defuzzificação descrito anteriormente na “rede de Petri possibilística” da Figura 3.8, uma possível sequência de eventos é descrita abaixo:

- t_3 foi pseudo-disparada anteriormente ($\eta(t_3) = incerta$)
 - $LL \leftarrow \{P_4\} \cup \{P_7\} = \{P_4, P_7\}$;
 - o disparo de t_3 é cancelado, ou seja, a ficha $\langle o_2, o_3 \rangle$ é removida do lugar P_7 e mantida no lugar P_4 (Figura 3.9(a)).
- $LL \neq \{\}$ então $p \leftarrow P_4$
 - a transição t_1 pertence ao conjunto das transições precedentes de P_4 e foi pseudo-disparada ($\eta(t_1) = incerta$),
 - * $LL \leftarrow \{P_7\} \cup \{P_1, P_2\} = \{P_7, P_1, P_2\}$;

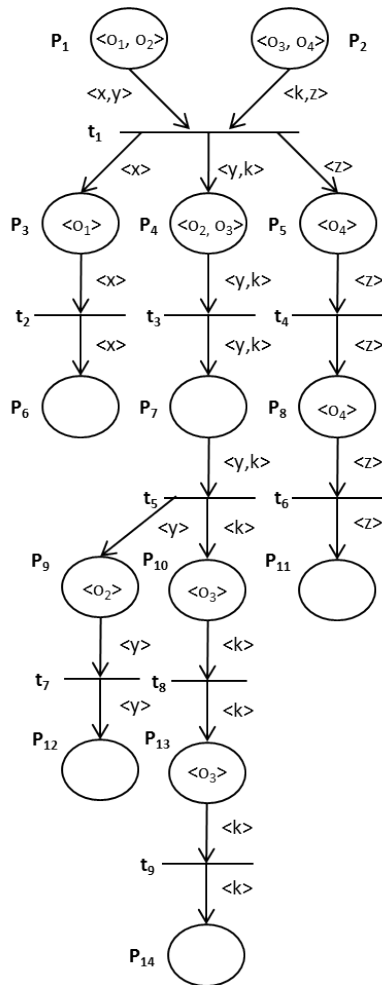
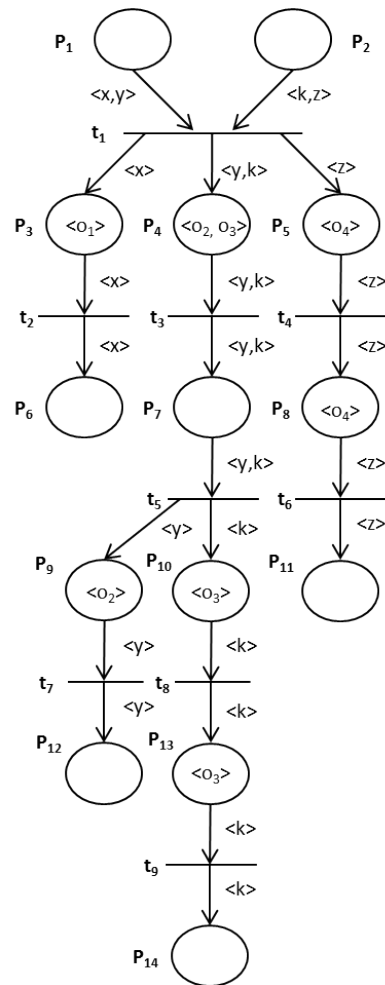
(a) cancelamento do disparo de t_3 (b) arquivamento do disparo de t_1

FIGURA 3.9: Aplicação do algoritmo de defuzzificação.

- * o disparo de t_1 é arquivado, ou seja, as fichas $\langle o_1, o_2 \rangle$ e $\langle o_3, o_4 \rangle$ são removidas, respectivamente, dos lugares P_1 e P_2 e as fichas $\langle o_1 \rangle$, $\langle o_2, o_3 \rangle$ e $\langle o_4 \rangle$ são mantidas, respectivamente, nos lugares P_3 , P_4 e P_5 (Figura 3.9(b)).
- não existe nenhuma transição pertencente ao conjunto das transições seguintes de P_4 que foram pseudo-disparadas.
- $LL \neq \{\}$ então $p \leftarrow P_7$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes de P_7 que foram pseudo-disparadas;
 - a transição t_5 pertence ao conjunto das transições seguintes de P_7 e foi pseudo-disparada ($\eta(t_5) = incerta$),
 - * $LL \leftarrow \{P_1, P_2\} \cup \{P_9, P_{10}\} = \{P_1, P_2, P_9, P_{10}\}$;

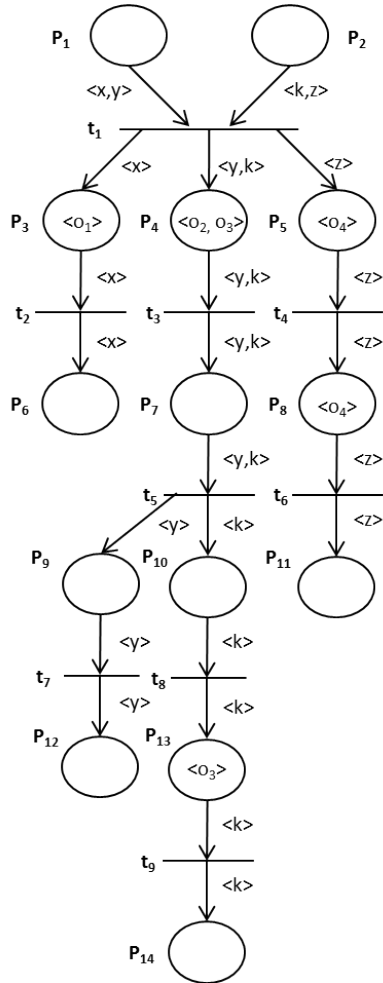
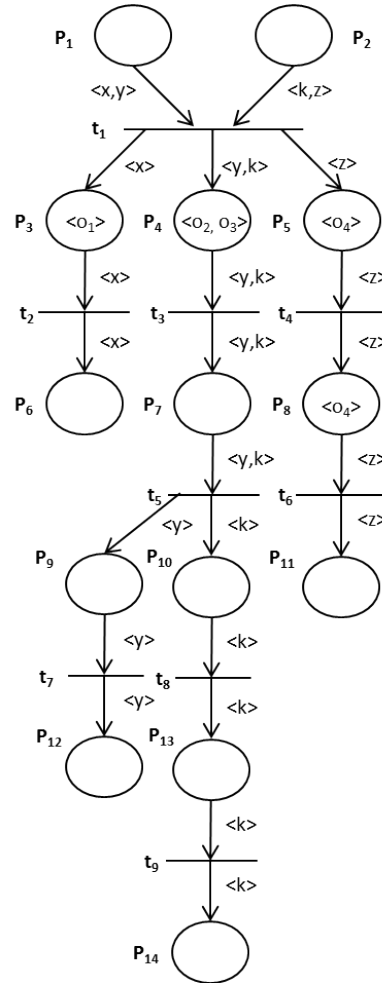
(a) cancelamento do disparo de t_5 (b) cancelamento do disparo de t_8

FIGURA 3.10: Continuação da aplicação do algoritmo de defuzzificação.

* o disparo de t_5 é cancelado, ou seja, as fichas $\langle o_2 \rangle$ e $\langle o_3 \rangle$ são removidas, respectivamente, dos lugares P_9 e P_{10} (Figura 3.10(a)).

- $LL \neq \{\}$ então $p \leftarrow P_1$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes e seguintes de P_1 que foram pseudo-disparadas.
- $LL \neq \{\}$ então $p \leftarrow P_2$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes e seguintes de P_2 que foram pseudo-disparadas.
- $LL \neq \{\}$ então $p \leftarrow P_9$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes e seguintes de P_9 que foram pseudo-disparadas.
- $LL \neq \{\}$ então $p \leftarrow P_{10}$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes de P_{10} que foram pseudo-disparadas;
 - a transição t_8 pertence ao conjunto das transições seguintes de P_{10} e foi pseudo-disparada ($\eta(t_8) = incerta$),
 - * $LL \leftarrow \{\} \cup \{P_{13}\} = \{P_{13}\}$;
 - * o disparo de t_8 é cancelado, ou seja, a ficha $\langle o_3 \rangle$ é removida do lugar P_{13} (Figura 3.10(b)).
- $LL \neq \{\}$ então $p \leftarrow P_{13}$
 - não existe nenhuma transição pertencente ao conjunto das transições precedentes e seguintes de P_{13} que foram pseudo-disparadas.
- $LL = \{\}$ então a transição t_3 é disparada normalmente produzindo a tupla $\langle o_2, o_3 \rangle$ no lugar P_7 e removendo-a do lugar P_4 (Figura 3.11).

Após a execução do algoritmo de defuzzificação, o posicionamento dos objetos nos lugares relacionados ao evento “ $\eta(t_3) = verdadeira$ ” torna-se certa. Entretanto, pode-se notar que o posicionamento do objeto $\langle o_4 \rangle$ continua incerto, isso ocorre porque a informação recebida não permite deduzir se o disparo da transição t_4 foi ou não válido. Para cancelar ou arquivar tal disparo um outro evento deverá ocorrer, ou “ $\eta(t_4) = verdadeira$ ” ou “ $\eta(t_6) = verdadeira$ ”.

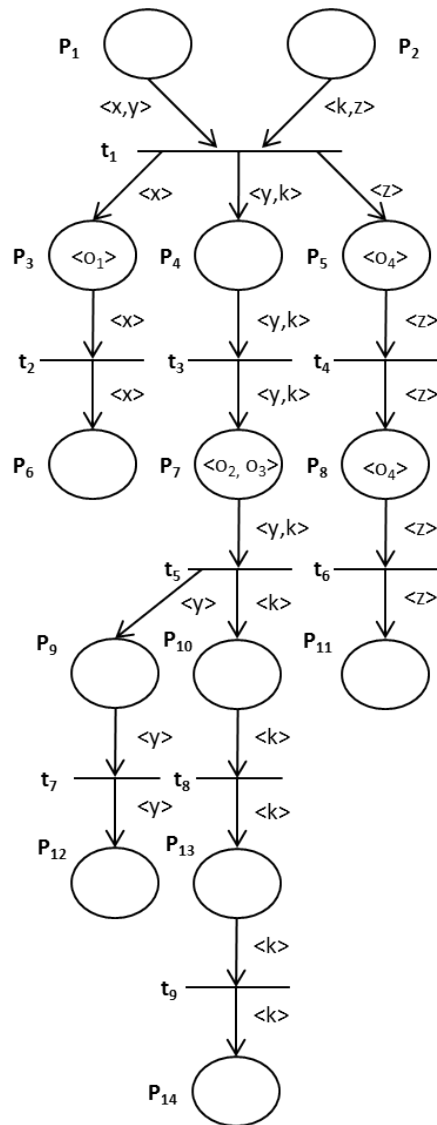


FIGURA 3.11: Disparo da transição t_3 , como na “rede de Petri clássica”, após a execução do algoritmo de defuzzificação.

3.4 *Workflow nets*

Uma rede de Petri que modela um processo de *workflow* é uma *Workflow net* [2, 14]. Uma *Workflow net* satisfaz as seguintes propriedades [2]:

- tem apenas um lugar de início (denominado *Start*) e apenas um lugar de término (denominado *End*), sendo estes dois lugares tratados como lugares especiais; o lugar *Start* tem apenas arcos de saída e o lugar *End* apenas arcos de entrada;
- uma ficha em *Start* representa um caso que precisa ser tratado e uma ficha em *End* representa um caso que já foi tratado;

- toda tarefa t (transição) e condição p (lugar) devem estar em um caminho que se encontra entre o lugar $Start$ e o lugar End .

A Figura 3.12 mostra os elementos de modelagem das *WorkFlow nets*.

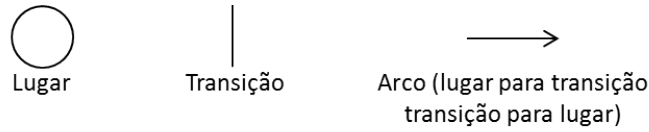


FIGURA 3.12: Elementos de Modelagem de uma *WorkFlow net*.

3.4.1 Processos

Um processo define quais tarefas precisam ser executadas e em qual ordem a execução deve ocorrer. Modelar um processo de *workflow* em termos de uma *WorkFlow net* é bem direto: transições são componentes ativos e modelam as tarefas, lugares são componentes passivos e modelam as condições (pré e pós) e as fichas modelam os casos [2].

Para ilustrar o mapeamento de processos em *WorkFlow nets*, considera-se o processo de tratamento de reclamações apresentado em [14]: “Uma reclamação é inicialmente gravada. Então, o cliente que efetuou a reclamação e o departamento responsável pela reclamação são contactados. O cliente é questionado para maiores informações. O departamento é informado sobre a reclamação. Estas duas tarefas podem ser executadas em paralelo, isto é, simultaneamente ou em qualquer ordem. Depois disso, os dados são recolhidos e uma decisão é tomada. Dependendo da decisão, ou um pagamento de compensação é efetuado, ou uma carta é enviada. Finalmente, a reclamação é armazenada”. A Figura 3.13 mostra a *WorkFlow net* que representa este processo.

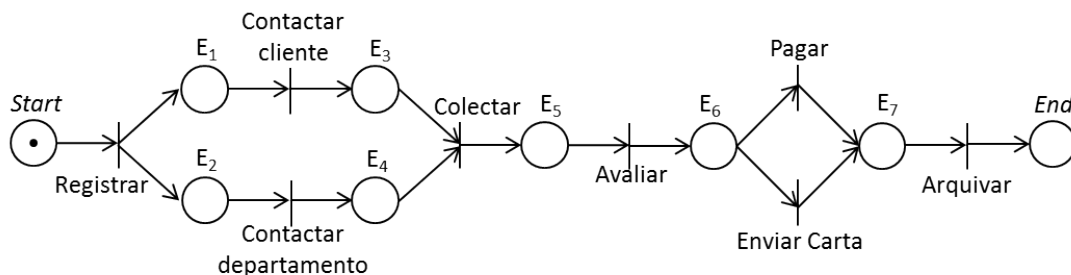


FIGURA 3.13: *WorkFlow net* para o processo de tratamento de reclamações e os seus acionamentos.

3.4.2 Roteamentos

De acordo com Aalst em [14], pela rota de um caso, ao longo de uma série de tarefas, pode-se determinar quais tarefas precisam ser executadas (e em que ordem). Quatro construções básicas para o roteamento de tarefas são consideradas:

- *sequencial*: a forma mais simples de execução de tarefas, onde uma tarefa é executada após a outra, havendo, claramente, dependência entre elas;
- *paralela*: mais de uma tarefa pode ser executada simultaneamente, ou em qualquer ordem. Neste caso, as tarefas podem ser executadas sem que o resultado de uma interfira no resultado das outras;
- *condicional (ou rota seletiva)*: quando há uma escolha entre duas ou mais tarefas;
- *iterativa*: quando é necessário executar uma mesma tarefa múltiplas vezes.

Considerando o processo de tratamento de reclamações, mostrado na Figura 3.13, as tarefas “*Contactar Cliente*” e “*Contactar Departamento*” são um exemplo de roteamento paralelo, as tarefas “*Colectar*” e “*Avaliar*” são um exemplo de roteamento sequencial e as tarefas “*Pagar*” e “*Enviar Carta*” são um exemplo de roteamento condicional.

O roteamento de um processo de *workflow* pode ser representado por uma rota iterativa, como descrito acima. Isso significa que uma certa atividade precisa ser executada repetidamente até que o resultado de um teste subsequente seja positivo.

Na abordagem proposta, uma rota iterativa é substituída por uma tarefa global, como mostrado na Figura 3.14. Na prática, se um processo de *workflow* deve respeitar um limite de tempo, ele não poderá repetir indefinidamente uma mesma atividade.

A estrutura hierárquica das redes de Petri, baseada no conceito de “blocos bem formados”³ [51], pode ser utilizada para representar uma rota iterativa através de uma única tarefa. Assim, uma duração máxima é associada a esta tarefa, a fim de especificar, de modo implícito, o número de vezes que a tarefa poderá ser executada dentro do bloco antes de ser detectado um problema na execução desta atividade.

³do inglês “Well-formed block”

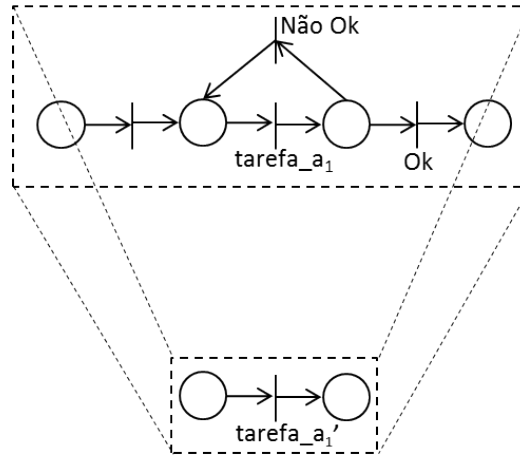


FIGURA 3.14: Bloco bem formado.

3.4.3 Acionamentos

Um acionamento é uma condição externa que guia a execução de uma tarefa sensibilizada [2]. Há quatro tipos distintos de tarefas:

- *usuário*: uma tarefa é acionada por um recurso humano e este acionamento é mostrado em uma *WorkFlow net* através do símbolo \Downarrow nas transições;
- *mensagem*: um evento externo aciona uma tarefa sensibilizada, o que é mostrado em uma *WorkFlow net* através do símbolo \boxtimes nas transições;
- *tempo*: uma tarefa sensibilizada é acionada por um relógio, isto é, a tarefa é executada em um tempo pré-definido. Isto é mostrado em uma *WorkFlow net* através do símbolo \odot nas transições;
- *automática*: uma tarefa é acionada no momento em que é sensibilizada e não requer interação humana. Para este tipo de tarefa não há nenhuma representação na *WorkFlow net*.

A Figura 3.15 mostra como cada um dos quatro tipos distintos de tarefas é associado às transições.

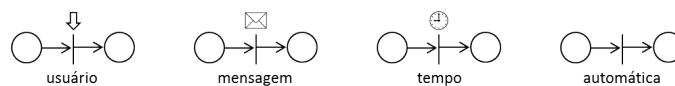


FIGURA 3.15: Tipos de tarefas nos acionamentos.

Nota-se que quando uma tarefa do tipo “usuário” é considerada, essa tarefa é acionada por um recurso humano, isto é, há uma alocação de recurso humano associada a esta tarefa. Nos demais tipos de tarefa não há alocação de recursos humanos associada.

O processo de tratamento de reclamações mostrado na Figura 3.13 consiste de oito tarefas, das quais três são automaticamente tratadas (*Registrar*, *Colectar*, *Arquivar*) e cinco são acionadas por recursos humanos (*Contactar Cliente*, *Contactar Departamento*, *Avaliar*, *Pagar* e *Enviar Carta*).

3.4.4 Soundness

Soundness [14] é o principal critério de corretude definido para *WorkFlow nets*. Uma *WorkFlow net* é *sound* se, e somente se, os três requisitos abaixo são satisfeitos:

- para cada ficha colocada no lugar de início, uma (e apenas uma) ficha aparecerá no lugar de término;
- quando uma ficha aparece no lugar de término, todos os outros lugares estão vazios, considerando o caso em questão;
- considerando uma tarefa associada à uma transição, é possível evoluir da marcação inicial até uma marcação que sensibiliza tal transição, ou seja, não deve haver nenhuma transição morta na *WorkFlow net*.

A propriedade *soundness* é um critério importante a ser satisfeito quando se trata de processos de negócios e a prova desse critério está relacionada com a análise qualitativa, no contexto das *WorkFlow nets*. Em [52, 53], métodos são apresentados para provar a propriedade *soundness*.

3.4.5 Monitoramento de Processos

Em [3], as *WorkFlow nets* foram revisadas em relação à sua capacidade para a monitoração dos processos de negócios. Os autores mostraram que alguns padrões não foram facilmente capturados, em particular, os padrões que tratam de cancelamentos e de várias instâncias de uma mesma tarefa sendo executadas simultaneamente.

A principal razão dessas limitações existente nas *WorkFlow nets* para a monitoração dos processos de negócios é o fato de que as tarefas são associadas diretamente às transições

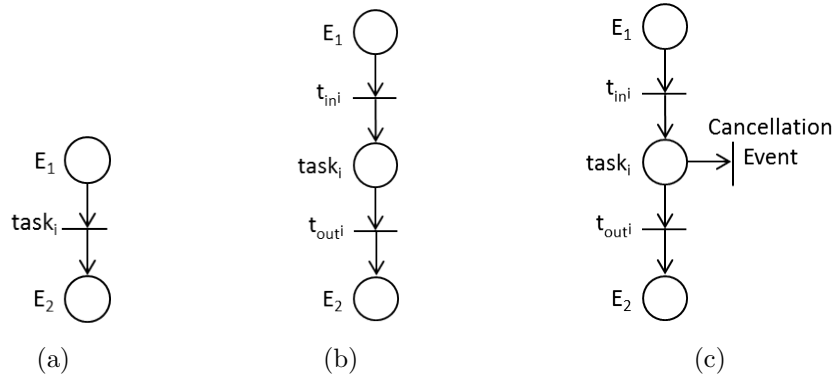


FIGURA 3.16: (a) *WorkFlow net* tradicional; (b) *WorkFlow net* com a execução explícita da tarefa; e (c) *WorkFlow net* com evento de cancelamento

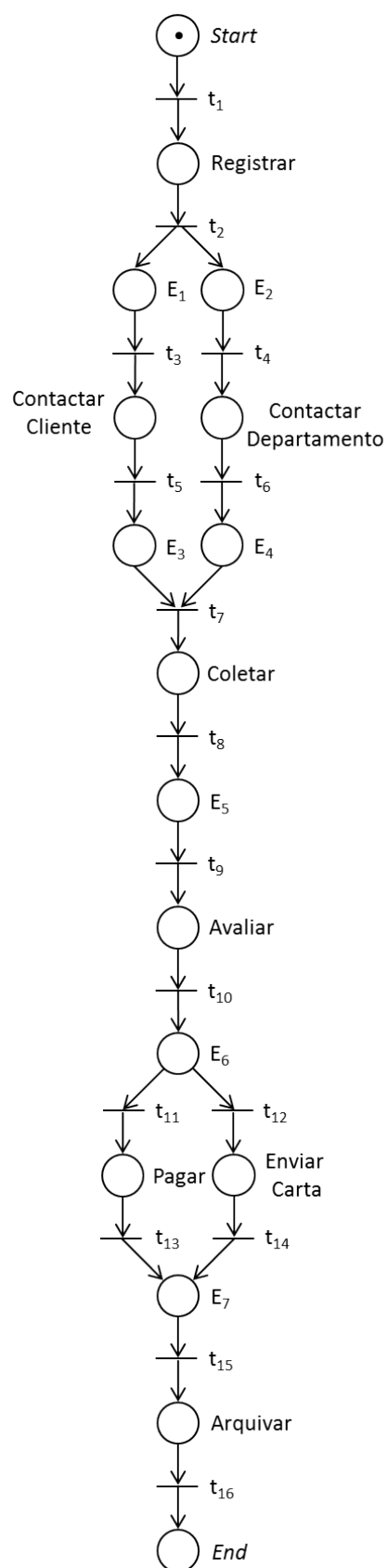
simples. Em consequência, uma vez iniciada, a tarefa não pode ser interrompida, pois isto corresponde ao disparo de uma transição.

Se durante a execução de uma tarefa, um evento ocorre no sistema, cuja finalidade é a de interromper o processo como um todo, nas *WorkFlow nets* tradicionais a execução das tarefas do processo têm de ser completadas inicialmente para, depois, serem capazes de aceitarem o cancelamento. É claro que um modelo adequado do processo deve ser capaz de aceitar a interrupção de eventos de um modo assíncrono, a fim de monitorar o processo de uma maneira eficiente.

A solução proposta em [43, 54, 55] para lidar com a interrupção durante a execução das tarefas é transformar as transições das *WorkFlow nets* em uma estrutura baseada no seguinte padrão: um bloco que corresponde à tarefa da transição t_i , composto de um lugar P_{t_i} , que representa o estado da tarefa t_i em execução, uma transição de entrada t_{ini} , que representa o início da execução da tarefa, e uma transição de saída t_{outi} , que representa o fim da execução da tarefa.

A *WorkFlow net* da Figura 3.16(a) é transformada num processo de *workflow* dado pelo modelo da rede de Petri acíclica da Figura 3.16(b). Como o novo bloco (correspondente à tarefa em execução) pode ser substituído por uma transição simples, preservando as boas propriedades do modelo inicial [15], o novo modelo do processo continuará *sound* e será adaptado para as atividades de monitoração, em particular, se alguns eventos de cancelamento precisam ser especificados como mostrado na Figura 3.16(c).

Finalmente, o modelo da *WorkFlow net* da Figura 3.17 é produzido a partir do modelo da Figura 3.13.

FIGURA 3.17: *Workflow net* com a execução explícita das tarefas.

3.5 Considerações Finais do Capítulo

Este capítulo apresentou os conceitos relacionados às redes de Petri e às *WorkFlow nets* necessários para o entendimento deste trabalho, definindo uma “rede de Petri clássica”, uma “rede de Petri a Objetos”, uma “rede de Petri possibilística” e o algoritmo de defuzzificação, além das *WorkFlow nets* e da propriedade “*soundness*”.

Capítulo 4

Workflow net Possibilística

A crescente complexidade dos processos, nas diversas organizações, estimula a adoção de técnicas de Gerenciamento de Processos de Negócios. Normalmente, os modelos dos processos estão na base destas técnicas e, geralmente, é feita a suposição de que esses modelos estão de acordo com a execução do processo em tempo real. No entanto, experiências recentes têm mostrado que isso muitas vezes não é o caso e, mesmo nos processos automatizados, desvios podem ocorrer [56]. Em outros casos, é desejável dispor de modelos que permitam flexibilidade [57].

Atualmente, os modelos não são mais utilizados apenas como instrumentos para descrever os processos existentes, eles tornaram-se uma parte integrante do processo de otimização, monitoração, e até mesmo de auditoria [58].

Na rede de Petri que é usada como um modelo para os processos de negócios num Sistema de Gerenciamento de *Workflow*, as mudanças de estado do processo são representadas pelas transições. Em particular, cada evento que ocorre durante a execução do processo (início e fim das atividades) será associado a uma transição como uma variável booleana¹. Tal variável é vista essencialmente como um valor externo correspondente a uma mensagem recebida de uma atividade (ou enviada a uma atividade). Possivelmente, os valores internos dos atributos das fichas também podem habilitar algumas transições. Tal rede de Petri pode ser executada diretamente por um mecanismo de inferência especializado chamado “jogador” de “rede de Petri clássica interpretada” [46], como o mostrado no diagrama de atividades na Figura 4.1, o qual permite a monitoração simplificada dos processos representados.

¹Um tipo de dado primitivo que possui dois valores, que podem ser considerados como 0 ou 1, verdadeiro ou falso.

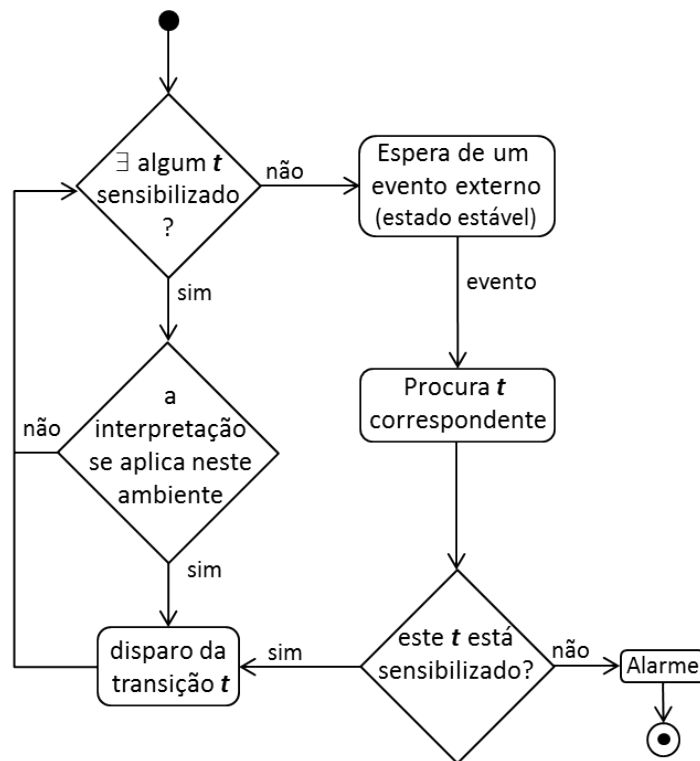


FIGURA 4.1: “Jogador” de “rede de Petri clássica interpretada”.

Entretanto, como indicado na introdução, a interação do comportamento humano no gerenciamento dos processos pode introduzir algumas incertezas na execução dos processos e estas devem ser consideradas no modelo destes a fim de tornar o modelo do processo mais robusto ao comportamento humano. O funcionamento do “jogador” mostrado na Figura 4.1 é baseado apenas na ocorrência normal dos eventos esperados. Se um evento inesperado ocorre, uma inconsistência imediata entre o modelo do processo e a execução em tempo real do mesmo é estabelecida, como por exemplo, a não ocorrência de um evento esperado levando o modelo a uma situação de bloqueio².

A abordagem proposta neste trabalho é baseada na combinação da estrutura de roteamento das *WorkFlow nets* com a marcação e o disparo impreciso das “redes de Petri possibilísticas”. Esse modelo produz um tipo de *WorkFlow net* possibilística que é capaz de lidar com os problemas de não conformidade na monitoração dos Processos de Negócio, permitindo a detecção e o tratamento de possíveis desvios ocasionados pelos atores humanos na execução do processo.

A definição da *WorkFlow net* possibilística é a seguinte:

Definição 4.1. (*WorkFlow net* possibilística) Uma *WorkFlow net* possibilística pode ser definida pela 9-upla:

²do inglês “deadlock” [59, 60, 61].

$$R = \langle P, T, C_{aso}, V, Pré, Pós, A_{tc}, A_{ta}, M_0 \rangle$$

onde:

- C_{aso} representa a classe de objeto “*Caso*”, onde um conjunto de atributos é definido e eventualmente organizado em uma hierarquia;
- P é um conjunto finito de lugares (o lugar de entrada *Start* (Início), o lugar de saída *End* (Fim) e os lugares envolvidos nos diversos roteiros da *WorkFlow net*), todos do tipo “*Caso*”;
- T é um conjunto finito de transições;
- V é um conjunto de variáveis formais do tipo “*Caso*”;
- $Pré$ é a função lugar precedente que, a cada arco de entrada de uma transição, faz corresponder uma soma formal de elementos de V ;
- $Pós$ é a função lugar seguinte que, a cada arco de saída de uma transição, faz corresponder uma soma formal de elementos de V ;
- A_{tc} é uma aplicação que, para cada transição, associa uma função de autorização que envolve o conjunto de atributos e métodos dos objetos por meio das variáveis formais V associados aos arcos de entrada;
- A_{ta} é uma aplicação que, para cada transição, associa uma ação que envolve os atributos ou métodos das variáveis formais associados aos arcos de entrada permitindo modificar seus atributos específicos por meio da invocação de seus métodos;
- M_0 é a marcação inicial que associa, ao lugar *Start*, uma soma formal dos objetos do tipo “*Caso*” (n -uplas de instâncias da classe “*Caso*”);).

Definição 4.2. (Tipos de Marcação) A marcação da *WorkFlow net* possibilística pode ser precisa ou imprecisa. São definidas abaixo:

- *Marcação precisa:* cada objeto do tipo *Caso* está localizado em apenas um lugar, ou seja, a possibilidade do objeto o estar no lugar p é 1 ($\pi_o(p) = 1$) e em todos os outros lugares pertencentes ao modelo, diferentes de p , a possibilidade é 0 ($\forall p_i \neq p, \pi_o(p_i) = 0$);
- *Marcação Imprecisa:* um mesmo objeto do tipo *Caso* está em dois ou mais lugares com possibilidade igual a 1 ($M(o, p_i) = 1$ e $\exists p_j \neq p_i \mid M(o, p_j) = 1$).

Definição 4.3. (Tipos de Disparo) O disparo numa *WorkFlow net* possibilística pode ser certo ou incerto. A definição de cada um é descrito abaixo:

- *Disparo Certo*: os objetos do tipo *Caso* são adicionados nos lugares de saída da transição correspondente e removidos dos lugares de entrada da mesma, imediatamente;
- *Disparo Incerto* ou *Pseudo-disparo*: os objetos do tipo *Caso* são apenas adicionados nos lugares de saída da transição correspondente e não são removidos dos lugares de entrada da mesma por falta de certeza sobre a ocorrência ou não do evento.

Definição 4.4. (Função de Autorização) A interpretação da *WorkFlow net* possibilística é dada pela função de autorização definida por:

$$\eta_{x_1, \dots, x_n} : T \longrightarrow \{Falsa, Incerta, Verdadeira\} \quad (4.1)$$

onde x_1, \dots, x_n são as variáveis associadas aos arcos de entrada da transição t .

Definição 4.5. (Regras de Disparo) Considerando o_1, \dots, o_n uma possível substituição para x_1, \dots, x_n para disparar t , as regras de disparo são definidas abaixo:

- a interpretação associada à transição t é verdadeira e a mesma está habilitada por uma marcação precisa, então um disparo como na “rede de Petri clássica” (com remoção dos objetos dos lugares de entrada) ocorre;
- a interpretação associada à transição t é incerta, então um pseudo-disparo (sem remoção dos objetos dos lugares de entrada) ocorre, considerando que a transição esteja habilitada independentemente do tipo de marcação (precisa ou imprecisa);
- a interpretação associada à transição t é verdadeira e a mesma está habilitada por uma marcação imprecisa, então o algoritmo de defuzzificação, detalhado na Seção 3.3.1, é chamado e uma nova computação de distribuição de possibilidade dos objetos do tipo *Caso* envolvidos na marcação imprecisa é realizado a fim de voltar à marcação precisa. Em seguida, o disparo certo da transição t é realizado;
- a interpretação associada à transição t é verdadeira porém a mesma não está habilitada por marcação alguma, então procedimentos para recuperar o estado certo do sistema são realizados.

Para ilustrar o modelo proposto, o processo de um “Serviço de Reclamações” apresentado na Seção 3.4 é transformado numa *WorkFlow net* possibilística. A Figura 4.2 representa a *WorkFlow net* possibilística que modela este processo. Em cada transição são associadas condições e ações. As condições certas e incertas são representadas, respectivamente, pelo símbolo “C” e “I” e a ação pelo símbolo “A”. Elas foram definidas, a título de exemplificação, seguindo as restrições impostas na descrição do mesmo e considerando um tempo limite em cada tarefa. Os atributos e métodos usados nas condições ou ações são representadas, respectivamente, pela assinatura $\langle \text{nomeObjeto} \rangle . \langle \text{nomeAtributo} \rangle$ e $\langle \text{nomeObjeto} \rangle . \langle \text{nomeAção} \rangle ()$, onde $\langle \text{nomeObjeto} \rangle$, $\langle \text{nomeAtributo} \rangle$ e $\langle \text{nomeAção} \rangle$ é substituído, respectivamente, pelo nome dado ao objeto, ao atributo definido no objeto associado e à ação executada no momento.

A *WorkFlow net* possibilística contém o objeto $\langle c_1 \rangle$, as variáveis x , y e z e todos os lugares associados à classe “Caso”. As instâncias do objeto da classe “Caso” têm os seguintes atributos: *data*, *iniRegistro*, *fimRegistro*, *iniCC*, *fimCC*, *iniCD*, *fimCD*, *fimCo*, *iniAv*, *fimAv*, *iniPag*, *fimPag*, *iniEC*, *fimEC*, *iniArq* e *fimArq*; e as seguintes funções: *registrar()*, *conCli()*, *conDep()*, *coletar()*, *avaliar()*, *pagar()*, *envCt()* e *arq()*.

Os atributos *iniRegistro*, *iniCC*, *iniCD*, *iniAv*, *iniPag*, *iniEC* e *iniArq* são verdadeiros quando as atividades *Registrar*, *Contactar Cliente*, *Contactar Departamento*, *Avaliar*, *Pagar*, *Enviar Carta* e *Arquivar* estão prontas para serem executadas, respectivamente. Os atributos *fimRegistro*, *fimCC*, *fimCD*, *fimCo*, *fimAv*, *fimPag*, *fimEC*, *fimArq* são verdadeiros quando a execução das atividades *Registrar*, *Contactar Cliente*, *Contactar Departamento*, *Coletar*, *Avaliar*, *Pagar*, *Enviar Carta* e *Arquivar* são finalizadas, respectivamente. O elemento τ , presente na condição ou na ação das transições, representa o tempo corrente.

As funções presentes nas transições não serão detalhadas, elas são meramente ilustrativas. Neste processo existe mais funções e atributos associados à classe “Caso”, além de possíveis atribuições de valores aos objetos, mas elas são dispensáveis para o entendimento do modelo possibilístico.

A abordagem proposta foca, dentre os tipos de flexibilidade citados na Seção 2, na flexibilidade por desvio na perspectiva do fluxo de controle [27, 28]. Esta flexibilidade caracteriza-se pela capacidade da instância do objeto desviar-se a partir das alternativas de execução descritas no modelo do processo sem modificá-lo.

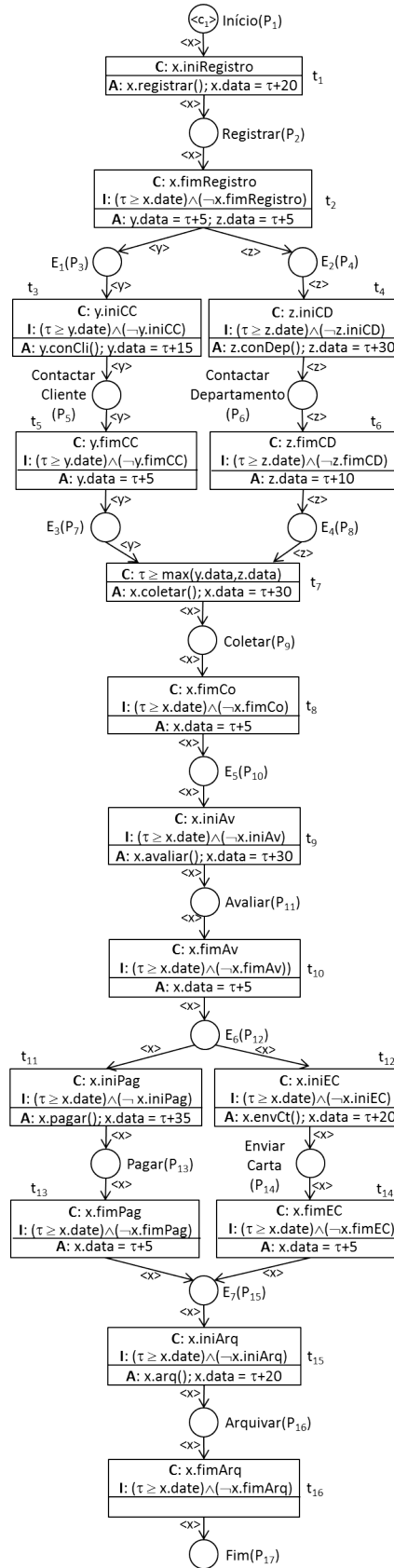


FIGURA 4.2: “Serviço de Reclamações”.

A Figura 4.3 ilustra, através das linhas finas, os diversos caminhos definidos no modelo para alcançar o fim da atividade a partir do seu início. Quando um desvio ocorre, um caminho alternativo é “criado”, apenas durante a execução do processo, para que o mesmo conclua sem que bloqueios surjam. Este caminho é representado na Figura 4.3 pela linha espessa associada à um dos caminhos definidos anteriormente.

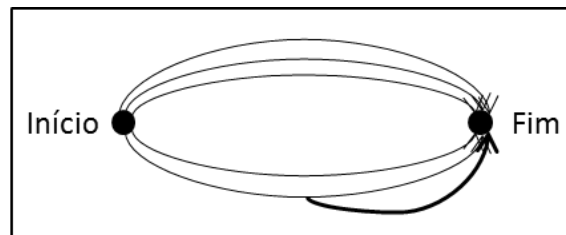


FIGURA 4.3: Flexibilidade por desvio.

A Figura 4.4 mostra um exemplo de desvio a partir do fluxo de controle especificado pelo modelo. O fluxo de controle do modelo apresentado na Figura 4.4(a) é especificado pela sequência das atividades *A*, *B* e *C*, ou seja, a atividade *B* deve ser realizada depois da atividade *A* e a atividade *C* depois da atividade *B*. Entretanto, neste tipo de flexibilidade, se um desvio ocorre durante a execução deste modelo, entende-se que é possível executá-lo de outras maneiras que não seja necessariamente a especificada, ou seja, diferente da execução das atividades *A*, *B* e *C* em sequência. Por exemplo, é possível, após a execução da atividade *A*, saltar *B* e executar diretamente *C*, como mostrado na Figura 4.4(b). Nota-se que no modelo correspondente na Figura 4.4(b) uma seta adicional representa o desvio, ou seja, representa a execução propriamente dita da instância.

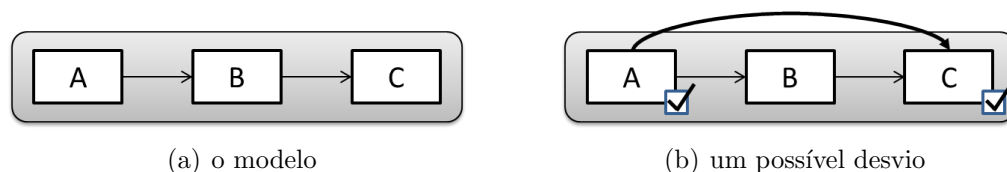


FIGURA 4.4: Flexibilidade por desvio na perspectiva do fluxo de controle.

Vários autores já propuseram abordagens para tratar este tipo de flexibilidade. Em [12, 41], a execução do processo é monitorada a fim de identificar a existência de desvios. Caso algum seja encontrado, o mesmo é informado ao responsável, o qual deve redirecionar a execução do processo a fim de obter uma execução consistente do mesmo em relação ao modelo. Em [33], o processo é definido por um conjunto de regras que definem quando o mesmo está ou não em conformidade com o modelo. Quando uma inconsistência é detectada, o processo prossegue até que uma informação torna

as regras verdadeiras. Entretanto, caso não seja possível, um procedimento para recuperar o estado do processo deve ser utilizado. Este último procedimento, porém não foi apresentado.

A abordagem proposta neste trabalho utiliza a incerteza gerada pelas marcações imprecisas para permitir um conjunto maior de possibilidades de execução no modelo sem alterá-lo. Para ilustrá-la, dois cenários são propostos. No primeiro apenas uma atividade é “saltada” e no segundo várias, entretanto o modelo não é alterado. O resultado obtido utilizando a abordagem possibilística é comparado quando se utiliza uma “rede de Petri clássica”. Estes cenários são detalhados nas Seções 4.1 e 4.2 respectivamente.

4.1 Cenário I - Desvio Fraco

No cenário I, apenas uma fração do processo “Serviço de Reclamação” é considerada. Esta fração é ilustrada na Figura 4.5. Nela considera-se que a reclamação já foi registrada e que novos objetos, idênticos à $\langle c_1 \rangle$ ($\langle c_{11} \rangle$ e $\langle c_{12} \rangle$), foram produzidos nos lugares E_1 e E_2 para que se possam iniciar simultaneamente as atividades “Contactar Cliente” (P_5) e “Contactar Departamento” (P_6).

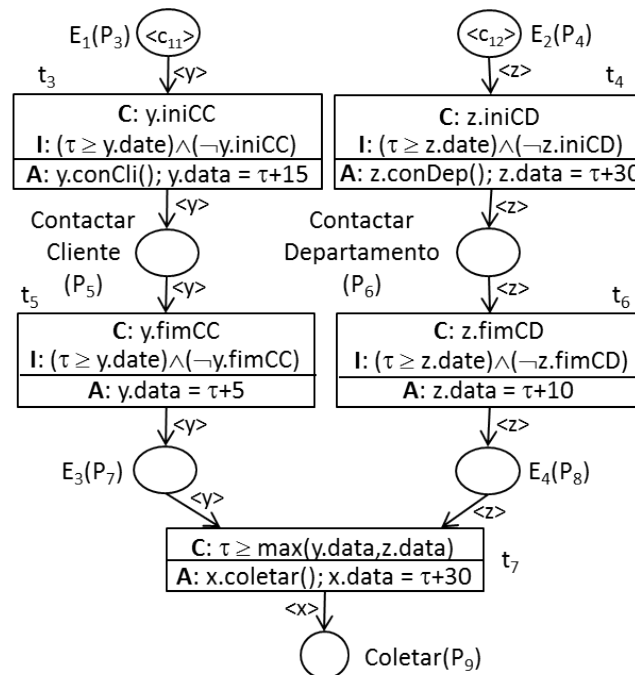


FIGURA 4.5: Fração do Processo de “Serviço de Reclamações”.

O comportamento esperado do processo é ter ambas atividades realizadas a fim de continuar o tratamento da reclamação através do disparo da transição t_7 , porém um desvio

pode eventualmente ocorrer. O recebimento das mensagens de término das atividades “*Contactar Cliente*” e “*Contactar Departamento*”, a partir do ator responsável, antes que o tempo corrente alcance os valores indicados pelo atributo *data* associado aos objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$, corresponde ao comportamento normal do processo.

Se as mensagens de término são recebidas antes do tempo determinado, todos os disparos serão certos e todas as marcações precisas. Porém, se o tempo corrente alcança o valor do atributo *data*, associado a um dos objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$, e nenhuma mensagem de término é recebida a partir do ator responsável para o objeto correspondente, alguns pseudo-disparos ocorrem e a imprecisão sobre alguns dos objetos aumenta.

Neste cenário é considerado que o ator responsável em entrar em contato com o cliente não conseguiu contatá-lo durante o tempo determinado pelo atributo *data* associado ao objeto $\langle c_{11} \rangle$. Normalmente, o propósito da atividade “*Contactar Cliente*” é coletar mais informações a partir do cliente para ajudá-lo a ter sucesso na resolução de sua reclamação. Logicamente, se ele não pode ser contatado, depois de um tempo pré-definido, o processo deve continuar, disparando a transição t_7 .

Para simular tal cenário, os seguintes eventos são considerados:

- as transições t_3 e t_4 serão disparadas de forma clássica no tempo $\tau = 10$ a efeito de ilustração;
- o atributo *fimCC* associado ao objeto c_{11} é *falso* durante todo o processo;
- o atributo *fimCD* associado ao objeto c_{12} é *verdadeiro* no tempo $\tau = 30$.

Inicialmente, antes de considerar o modelo possibilístico proposto, este cenário é simulado baseado no funcionamento do “jogador” descrito pela Figura 4.1, portanto as interpretações incertas não são consideradas neste momento. Abaixo essa simulação é detalhada:

- no tempo $\tau = 10$, as transições t_3 e t_4 são disparadas produzindo, respectivamente, o objeto $\langle c_{11} \rangle$ no lugar P_5 (Figura 4.6(a)) e o objeto $\langle c_{12} \rangle$ no lugar P_6 (Figura 4.6(b)). Ao realizar o disparo, as ações relacionadas as transições t_3 e t_4 são executadas, logo os atributos *data*, associados aos objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$, são atualizados, respectivamente, para os valores 25 e 40;
- no tempo $\tau = 30$, a transição t_6 é disparada, pois está habilitada e sua condição é verificada (Figura 4.6(c)). Ao realizar o disparo, a ação relacionada à transição

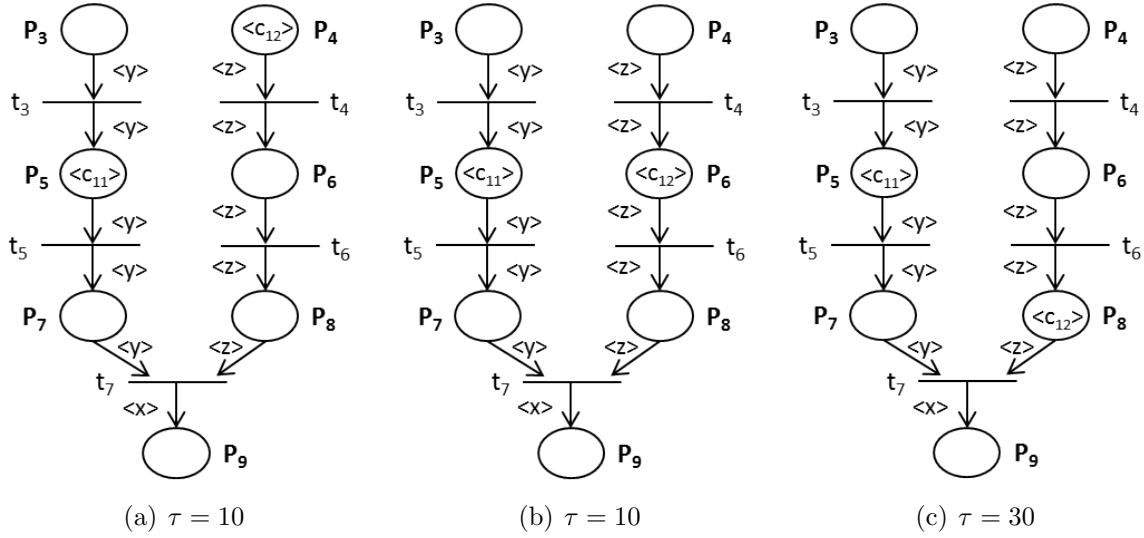


FIGURA 4.6: Resultados da simulação do cenário I considerando o “jogador” de “rede de Petri clássica interpretada”.

t_6 é executada, logo o valor do atributo *data*, associado ao objeto $\langle c_{12} \rangle$, é atualizado para 40;

- após o tempo $\tau > 30$, o processo entra num estado de bloqueio, isto é, nenhuma transição é habilitada para qualquer disparo, pois a condição associada à transição t_5 nunca será verificada ($fimCC$ é *falsa* em qualquer momento), impedindo o disparo da mesma e a continuação da execução do processo.

Para impedir esse bloqueio e permitir que o processo prossiga, disparos incertos e marcações imprecisas serão aplicados para a conclusão do mesmo. A simulação deste cenário, detalhado abaixo, é baseada no “jogador” de “rede de Petri possibilística” ilustrado na Figura 4.7.

- no tempo $\tau = 10$, as transições t_3 e t_4 são disparadas normalmente produzindo, respectivamente, o objeto $\langle c_{11} \rangle$ no lugar P_5 (Figura 4.8(a)) e o objeto $\langle c_{12} \rangle$ no lugar P_6 (Figura 4.8(b)). Ao realizar o disparo, as ações relacionadas às transições t_3 e t_4 são executadas, então o atributo *data*, associado aos objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$, é atualizado, respectivamente, para os valores 25 e 40;
- no tempo $\tau = 25$, a interpretação $\eta_{c_{11}}$ associada à transição t_5 é incerta ($(\tau \geq c_{11}.date) \wedge (\neg c_{11}.fimCC)$), logo a transição t_5 é pseudo-disparada. Uma cópia do objeto $\langle c_{11} \rangle$ é produzida no lugar P_7 (Figura 4.8(c)) e nenhuma ação é realizada neste tipo de disparo, pois não se tem certeza sobre a ocorrência ou não da conclusão da atividade “*Contactar Cliente*”;

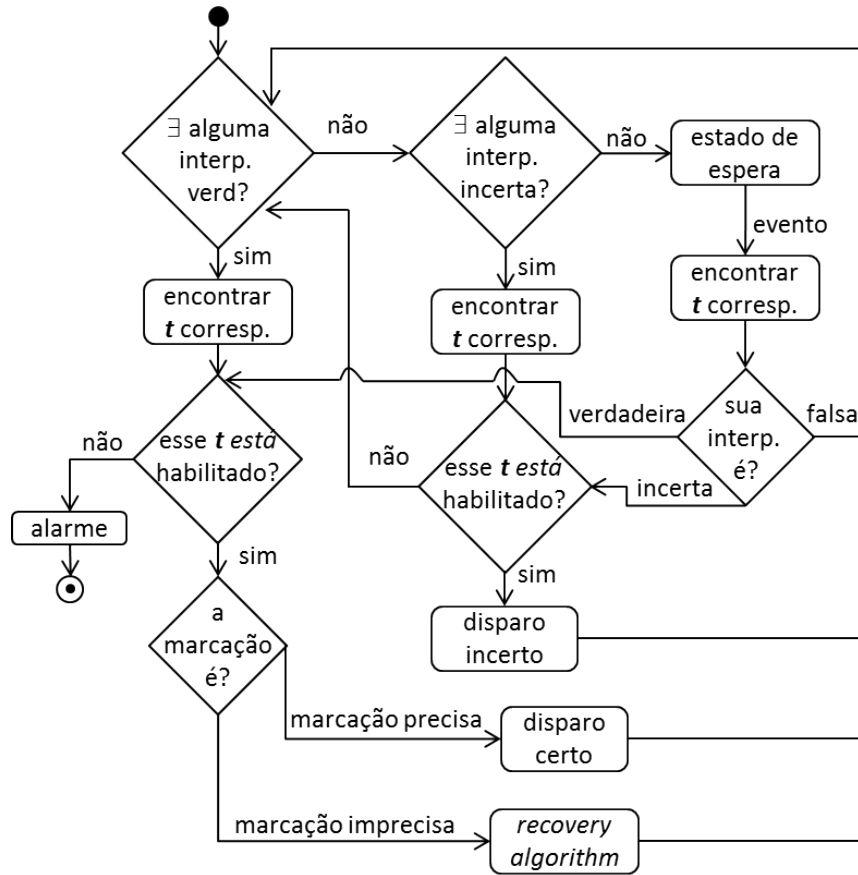


FIGURA 4.7: “Jogador” de “rede de Petri possibilística”.

- no tempo $\tau = 30$, a interpretação $\eta_{c_{12}}$ associada à transição t_6 é verdadeira, pois o atributo *fimCD* associado ao objeto c_{12} se torna verdadeiro, logo a transição t_6 é disparada como numa “rede de Petri clássica”(Figura 4.8(d)). Um novo objeto ($< c_{12} >$) é produzido no lugar P_8 e a ação associada à transição é realizada alterando o valor do atributo *data*, pertencente a este objeto, para 40;
- no tempo $\tau = 40$, a interpretação $\eta_{c_{11}c_{12}}$ associada à transição t_7 é verdadeira ($\tau \geq \max(y.data, z.data)$). Neste momento, a transição t_7 está habilitada por uma marcação imprecisa e uma interpretação verdadeira, consequentemente o algoritmo de defuzzificação, detalhado na Seção 3.3.1, para recuperar a certeza no processo é chamado. Após a execução do algoritmo, a marcação tornou-se precisa (Figura 4.8(e)), a transição t_7 é disparada normalmente produzindo um novo objeto ($< c_1 >$) no lugar P_9 e as ações relacionadas à esta transição são executadas (Figura 4.8(f)).

Este cenário corresponde a uma situação em que as limitações de tempo não permitem uma resposta do cliente. Mesmo assim, através dos disparos incertos, neste cenário nenhuma inconsistência é detectada e o processo prossegue normalmente. A Figura 4.9

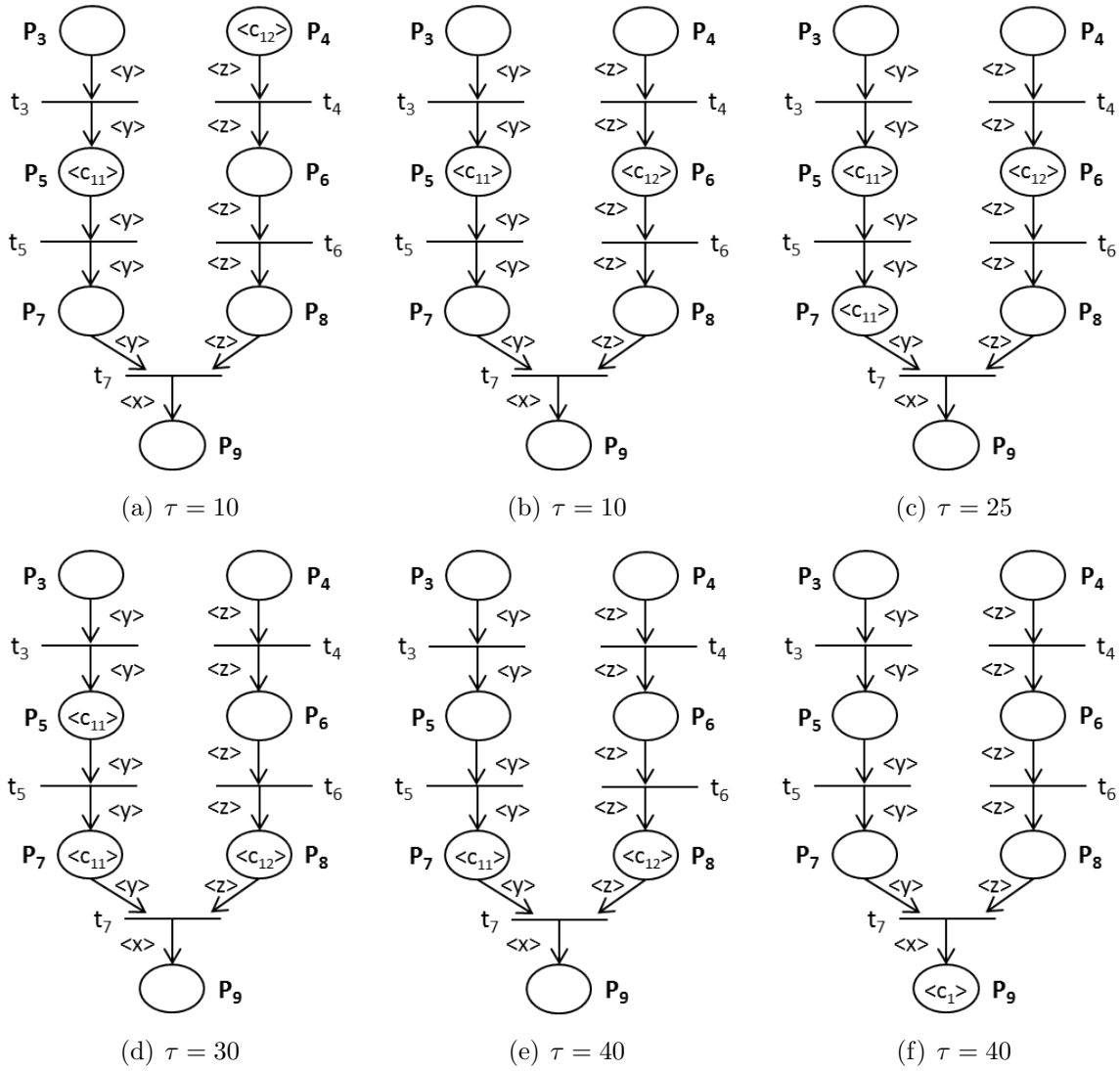


FIGURA 4.8: Resultados da simulação do cenário I.

mostra as distribuições de possibilidades das instâncias dos objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$ em função do tempo (as linhas grossas representam uma possibilidade igual a 1(um) e as linhas finas uma possibilidade igual a 0(zero)).

Para que o modelo do processo descreva este cenário sem que um desvio ocorra considerando uma *WorkFlow net* não possibilística, uma transição t deve ser adicionada no modelo de tal forma que a atividade “*Contactar Cliente*” pode ser ou não realizada. A Figura 4.10 mostra tal alteração na fração do processo, entretanto, diversas combinações podem ser adicionadas nesta fração a fim de considerar todas as alternativas possíveis de execução, porém, caso adicionadas, a legibilidade do modelo seria mais difícil.

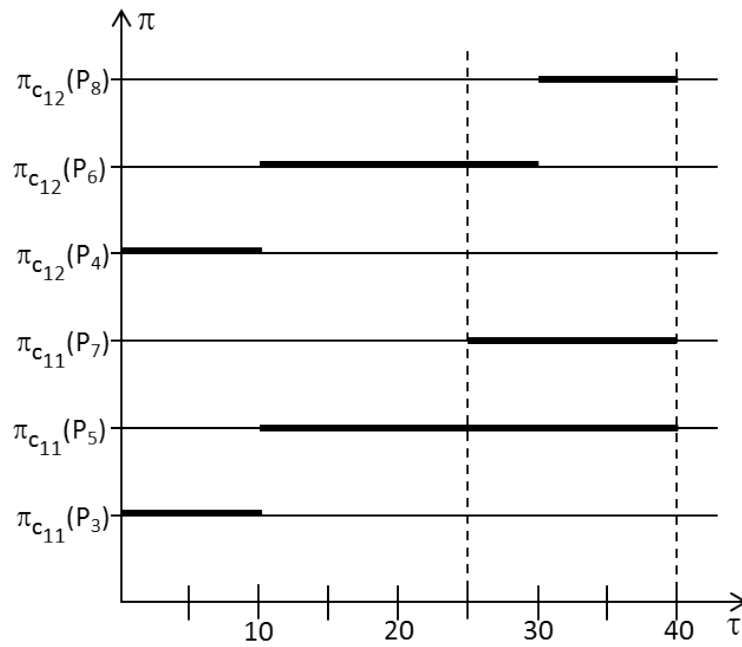


FIGURA 4.9: Distribuições das possibilidades relacionadas aos locais dos objetos c_{11} e c_{12} .

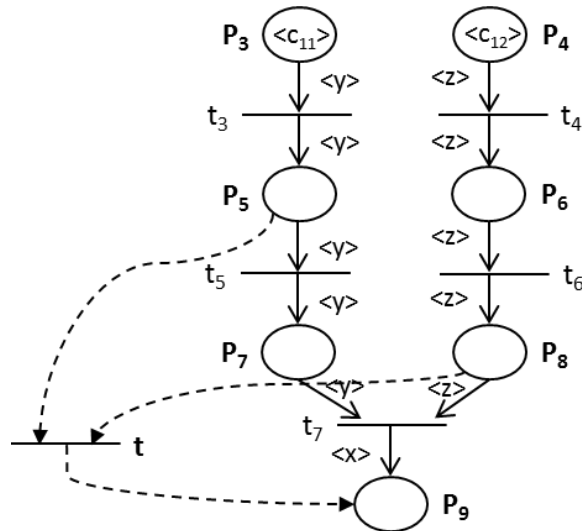


FIGURA 4.10: Fração do processo “Serviço de Reclamações” alterado para suportar o Cenário I.

4.2 Cenário II - Desvio Forte

No cenário II é considerado que a transição t_1 já foi disparada e que o método “*registrar()*” foi chamado e enviado para o ator correspondente (neste caso, o ator é uma secretária). Após isto, o objeto $\langle c_1 \rangle$ é produzido no lugar “*Registrar*” (P_2) (Figura 4.11). Posteriormente, é esperado que o fim do registro da reclamação (quando a condição

“*fimRegistro*”, associado à transição t_2 , torna-se verdadeira) seja informado para que o processo possa prosseguir normalmente.

Supondo que a secretária tenha um certo nível de autonomia em tomar decisões em função do seu conhecimento especializado, então, ela poderá determinar se a reclamação foi ou não devidamente apresentada. Por exemplo, ela pode decidir enviar uma carta diretamente para o proprietário da reclamação em vez de seguir fielmente o conjunto completo de atividades especificadas pelo modelo do processo.

Neste caso, após o envio de uma carta, em vez da interpretação associada com a transição t_2 ($c_1.fimRegistro$) tornar-se verdadeira, é a interpretação associada com a transição t_{14} ($c_1.fimEC$) que se torna verdadeira. Uma vez que nenhum objeto existe no lugar “*Enviar Carta*” (P_{14}) e a interpretação da transição t_2 é falsa, então, neste momento, nem a transição t_{14} nem a transição t_2 podem ser disparadas. Portanto o estado global do processo é de bloqueio.

Para tal situação, ao se aplicar o “jogador” descrito pela Figura 4.1 ou o “jogador” de “rede de Petri possibilística” da Figura 4.7, um alarme é disparado e uma recuperação do estado tem de ser realizada manualmente após o diagnostico do motivo da inconsistência do processo correspondente.

Desta forma, em vez de aplicar tais “jogadores”, um novo “jogador” de “rede de Petri possibilística”, dado pelo diagrama de atividades da Figura 4.12, é considerado. Neste caso é verificado se existe pelo menos uma possível sequência de pseudo-disparos que torna o estado global da “rede de Petri possibilística” consistente com os eventos externos. Esta sequência deve habilitar a transição que contém a sua interpretação verdadeira. Entretanto, caso esta sequência não seja encontrada, um alarme é disparado e o estado do processo deve ser recuperado manualmente.

A equação fundamental da teoria das redes de Petri [15] permite verificar se existe uma sequência não ordenada de disparo de transições que possibilita alcançar uma marcação tal que a transição que recebe o evento externo seja habilitada. Entretanto, deve-se verificar se tal sequência é uma condição necessária neste processo.

Neste cenário, conside a *WorkFlow net* da Figura 4.2 com o objeto $< c_1 >$ presente no lugar “*Registrar*” (P_2) (Figura 4.11) e, a transição t_{14} recebendo um evento externo que torna a condição “ $x.fimEC$ ”, associada à esta transição, verdadeira. Uma vez que a marcação corrente desta *WorkFlow net* não habilita o disparo da transição t_{14} e sua interpretação é verdadeira, os procedimentos para recuperar o estado consistente do modelo é realizado.

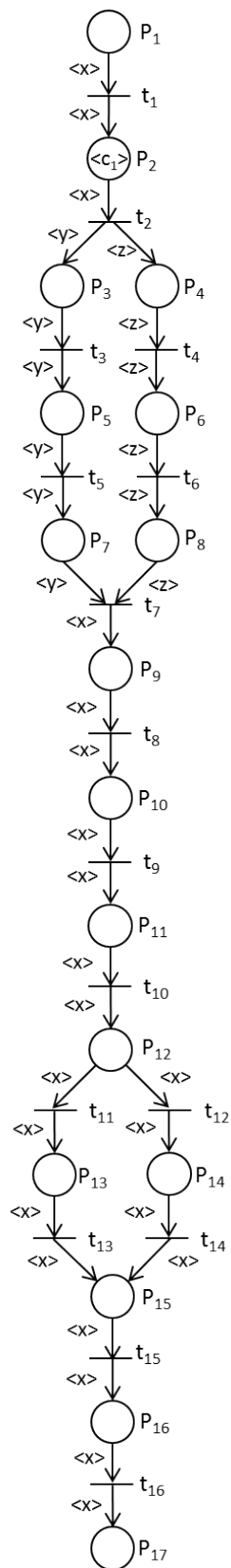


FIGURA 4.11: Estado inicial a ser considerado no cenário II.

Para resolver essa igualdade de matriz, um sistema linear é derivado e sua solução é apresentada abaixo:

$$\left\{ \begin{array}{lll} -x_1 = 0 & \Rightarrow & x_1 = 0 \\ x_1 - x_2 + 1 = 0 & \Rightarrow & x_2 = x_1 + 1 \Rightarrow x_2 = 1 \\ x_2 - x_3 = 0 & \Rightarrow & x_3 = x_2 \Rightarrow x_3 = 1 \\ x_2 - x_4 = 0 & \Rightarrow & x_4 = x_2 \Rightarrow x_4 = 1 \\ x_3 - x_5 = 0 & \Rightarrow & x_5 = x_3 \Rightarrow x_5 = 1 \\ x_4 - x_6 = 0 & \Rightarrow & x_6 = x_4 \Rightarrow x_6 = 1 \\ x_5 - x_7 = 0 & \Rightarrow & x_7 = x_5 \Rightarrow x_7 = 1 \\ x_6 - x_7 = 0 & & \\ x_7 - x_8 = 0 & \Rightarrow & x_8 = x_7 \Rightarrow x_8 = 1 \\ x_8 - x_9 = 0 & \Rightarrow & x_9 = x_8 \Rightarrow x_9 = 1 \\ x_9 - x_{10} = 0 & \Rightarrow & x_{10} = x_9 \Rightarrow x_{10} = 1 \\ x_{10} - x_{11} - x_{12} = 0 & \Rightarrow & x_{11} + x_{12} = x_{10} \Rightarrow x_{11} = 1 - x_{12} \\ x_{11} - x_{13} = 0 & \Rightarrow & x_{11} = x_{13} \Rightarrow x_{11} = -x_{14} \\ x_{12} - x_{14} = 1 & \Rightarrow & x_{12} = 1 + x_{14} \\ x_{13} + x_{14} - x_{15} = 0 & \Rightarrow & x_{13} + x_{14} = x_{15} \Rightarrow x_{13} = -x_{14} \\ x_{15} - x_{16} = 0 & \Rightarrow & x_{15} = x_{16} \Rightarrow x_{15} = 0 \\ x_{16} = 0 & & \end{array} \right.$$

As variáveis x_{11} , x_{12} e x_{13} são dadas em função da variável x_{14} . Como a sequência S é válida somente se for positiva, todos os valores pertencentes a S têm de ser positivos, ou seja, maior ou igual a zero. Com isso temos que:

$$\left\{ \begin{array}{lll} x_{11} \geq 0 & \Rightarrow & 1 - x_{12} \geq 0 \Rightarrow x_{12} \leq 1 \\ x_{12} \geq 0 & \Rightarrow & 1 + x_{14} \geq 0 \Rightarrow x_{14} \geq -1 \\ x_{13} \geq 0 & \Rightarrow & -x_{14} \geq 0 \Rightarrow x_{14} \leq 0 \end{array} \right.$$

Considerando o sistema linear anterior e que todas as variáveis têm de ser positivas, x_{14} deverá ser igual à 0 (zero) e, conseqüentemente, $x_{13} = 0$, $x_{12} = 1$ e $x_{11} = 0$. Logo, o vetor obtido da sequência não ordenada S , a partir da resolução da Equação 4.2, é: $S^t = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$, correspondendo à seguinte sequência não ordenada de transições: $t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}$.

Para qualquer rede de Petri, o resultado produzido pela sequência não ordenada S é apenas uma condição necessária no problema de alcançabilidade da teoria das redes de Petri [15, 47]. Entretanto, no caso da *WorkFlow net* que respeita a propriedade *sound*,

S também se torna uma condição suficiente, pois não existe qualquer parte morta na rede correspondente.

Como a *Workflow net* da Figura 4.2, que modela o processo de um “Serviço de Reclamações” utilizado neste cenário, respeita a propriedade *sound* e a sequência não ordenada S foi encontrada, faz-se necessário encontrar pelo menos uma sequência ordenada de disparo de transições que, ao dispará-las, a marcação final (M_f) será obtida a partir da marcação corrente (M_c).

O segundo passo para recuperar o estado consistente do processo é transformar, apenas durante o procedimento de recuperação, as interpretações associadas a todas as transições t_i , que pertencem à solução não nula de S , em *incertas* ($\eta_{var}(t_i) = incerta$)³ e todas as outras t_j , pertencentes à *Workflow net* possibilística (Figura 4.11), em *falsas* ($\eta_{var}(t_j) = falsa$), exceto a transição t_{14} que corresponde à transição com interpretação verdadeira ($\eta_{var}(t_{14}) = verdadeira$) por causa da ocorrência do evento externo. Em seguida, o pseudo-disparo destas transições é realizado para encontrar uma sequência ordenada de disparo (lembre-se que os pseudo-disparos de uma “rede de Petri possibilística” não executam as ações associadas às transições disparadas):

- a partir da marcação M_c , correspondente a um objeto $\langle c_1 \rangle$ em P_2 (Figura 4.11), a transição t_2 está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$ em P_3 ($\langle c_{11} \rangle$) e em P_4 ($\langle c_{12} \rangle$), sem remover o objeto de P_2 . A marcação imprecisa M_{int_1} resultante é dada pelas cópias de $\langle c_1 \rangle$ em P_2 , de $\langle c_{11} \rangle$ em P_3 e de $\langle c_{12} \rangle$ em P_4 (Figura 4.13(a));
- a partir da marcação M_{int_1} , as transições t_3 e t_4 estão pseudo habilitadas e elas podem ser pseudo disparadas. Se primeiramente a transição t_3 é escolhida, uma nova cópia do objeto $\langle c_{11} \rangle$ (o qual é cópia do objeto $\langle c_1 \rangle$) é produzida em P_5 . A marcação imprecisa M_{int_2} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , do $\langle c_{11} \rangle$ em P_3 e P_5 , e do $\langle c_{12} \rangle$ em P_4 (Figura 4.13(b));
- a partir da marcação M_{int_2} , as transições t_4 e t_5 estão pseudo habilitadas e elas podem ser pseudo disparadas. Se primeiramente a transição t_4 é escolhida, uma nova cópia do objeto $\langle c_{12} \rangle$ (o qual é cópia do objeto $\langle c_1 \rangle$) é produzida em P_6 . A marcação imprecisa M_{int_3} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , do $\langle c_{11} \rangle$ em P_3 e P_5 , e do $\langle c_{12} \rangle$ em P_4 e P_6 (Figura 4.13(c));
- a partir da marcação M_{int_3} , as transições t_5 e t_6 estão pseudo habilitadas e elas podem ser pseudo disparadas. Se primeiramente a transição t_5 é escolhida, uma

³*var* corresponde às variáveis associadas aos arcos de entrada da transição t_i

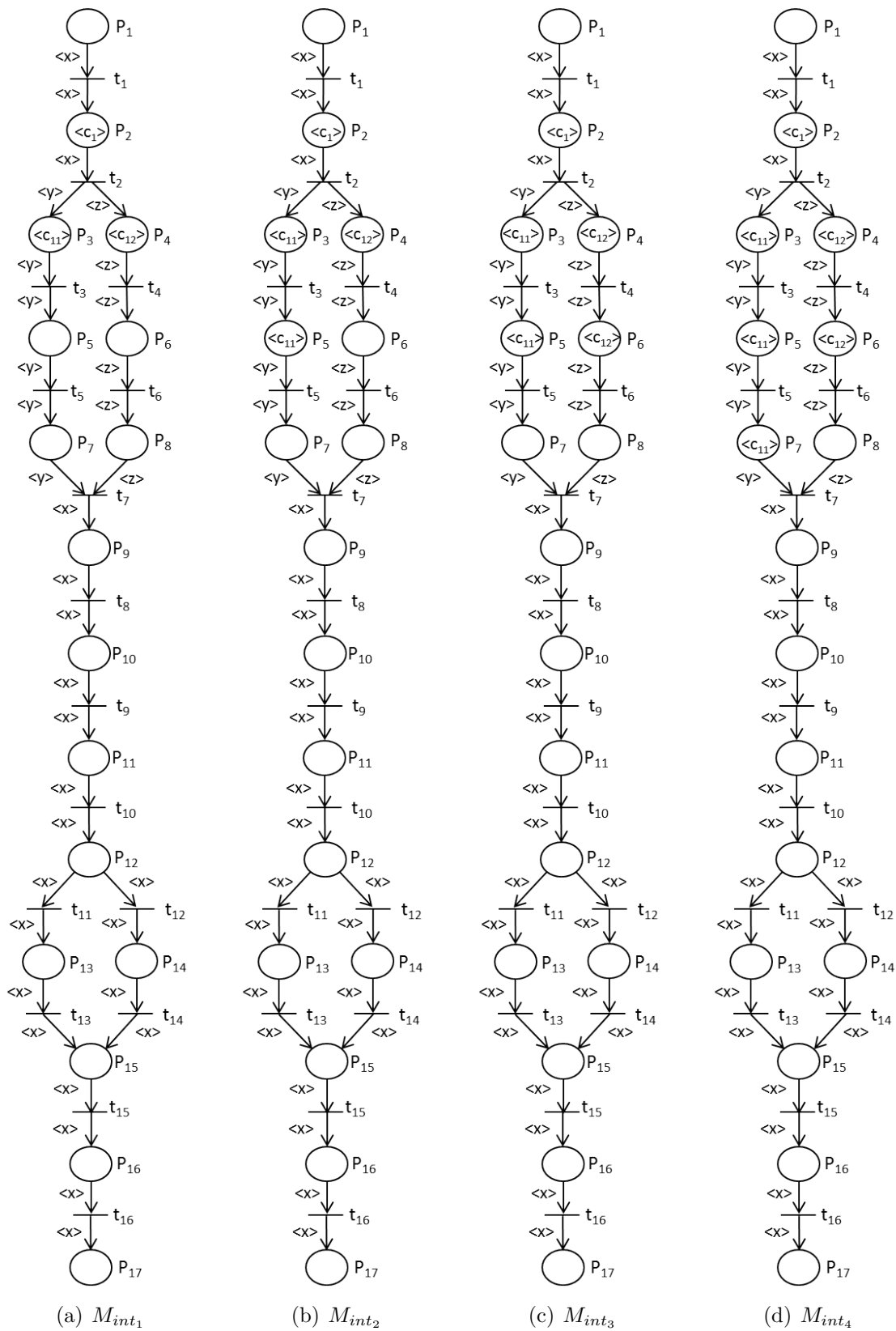


FIGURA 4.13: Resultados da simulação do cenário II.

nova cópia do objeto $\langle c_{11} \rangle$ é produzida em P_7 . A marcação imprecisa M_{int_4} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 e P_6 (Figura 4.13(d));

- a partir da marcação M_{int_4} , a transição t_6 está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_{12} \rangle$ em P_8 . A marcação imprecisa M_{int_5} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.14(a));
- a partir da marcação M_{int_5} , a transição t_7 está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$, dado que os objetos $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$ são cópias dele, em P_9 . A marcação imprecisa M_{int_6} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 e P_9 , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.14(b));
- a partir da marcação M_{int_6} , a transição t_8 está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$ em P_{10} . A marcação imprecisa M_{int_7} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , P_9 e P_{10} , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.14(c));
- a partir da marcação M_{int_7} , a transição t_9 está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$ em P_{11} . A marcação imprecisa M_{int_8} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , P_9 , P_{10} e P_{11} , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.14(d));
- a partir da marcação M_{int_8} , a transição t_{10} está pseudo habilitada e pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$ em P_{12} . A marcação imprecisa M_{int_9} resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , P_9 , P_{10} , P_{11} e P_{12} , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.15(a));
- a partir da marcação M_{int_9} , apenas a transição t_{12} está pseudo habilitada, pois a interpretação da transição t_{11} é falsa. Logo, ela pode ser pseudo disparada produzindo uma nova cópia do objeto $\langle c_1 \rangle$ em P_{14} . A marcação imprecisa $M_{int_{10}}$ resultante é dada pelas cópias do objeto $\langle c_1 \rangle$ em P_2 , P_9 , P_{10} , P_{11} , P_{12} e P_{14} , do $\langle c_{11} \rangle$ em P_3 , P_5 e P_7 , e do $\langle c_{12} \rangle$ em P_4 , P_6 e P_8 (Figura 4.15(b)).

Finalmente, a marcação final $M_{int_{10}}$, obtida através dos pseudo-disparos, correspondente à *WorkFlow net* possibilística da Figura 4.15(b), habilita a transição t_{14} (onde o evento externo foi detectado). Nesta marcação imprecisa, a possibilidade de ter uma cópia do mesmo objeto $\langle c_1 \rangle$ nos lugares P_2 , P_3 , P_4 , P_5 , P_6 , P_7 , P_8 , P_9 , P_{10} , P_{11} , P_{12} e

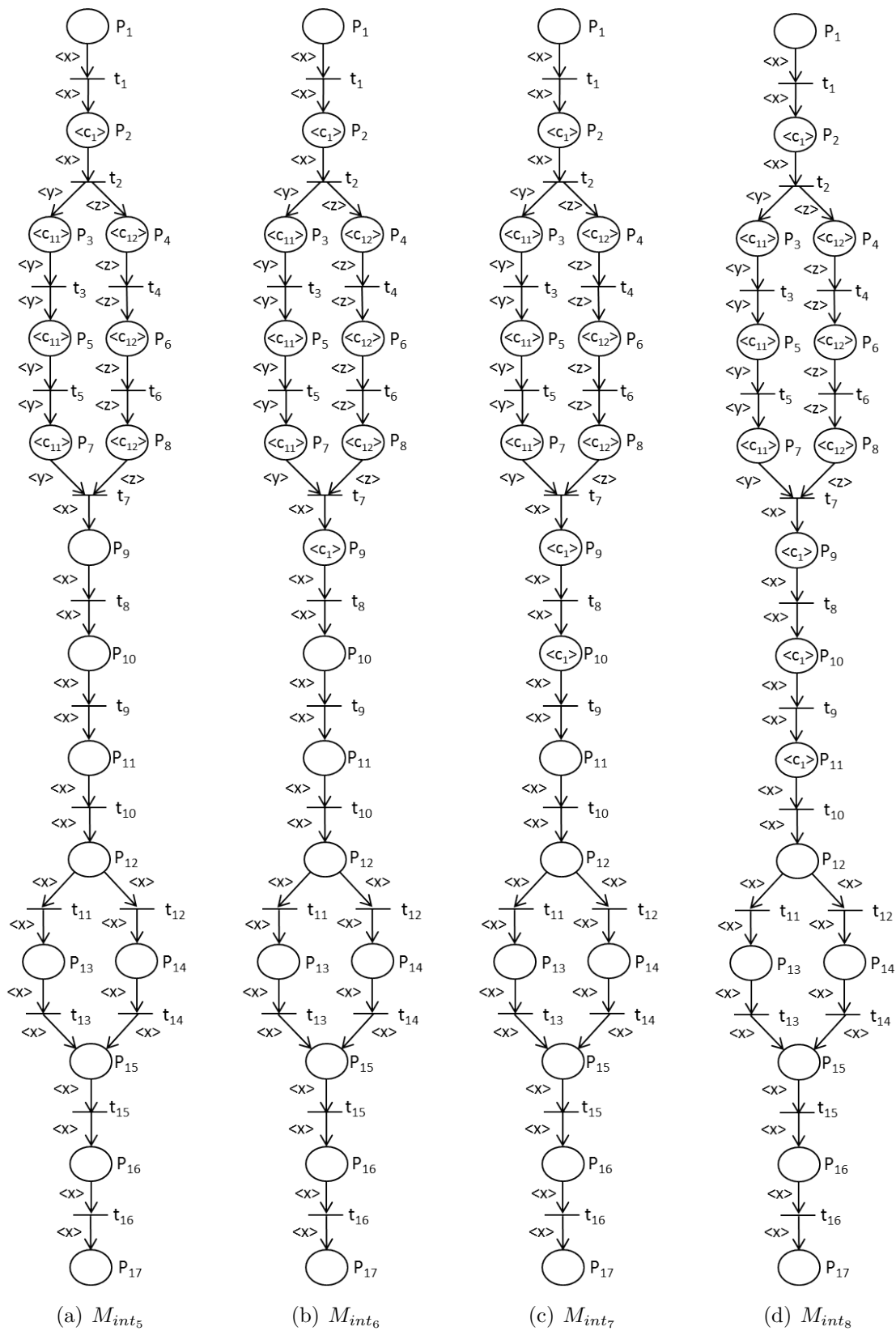


FIGURA 4.14: Resultados da simulação do cenário II.

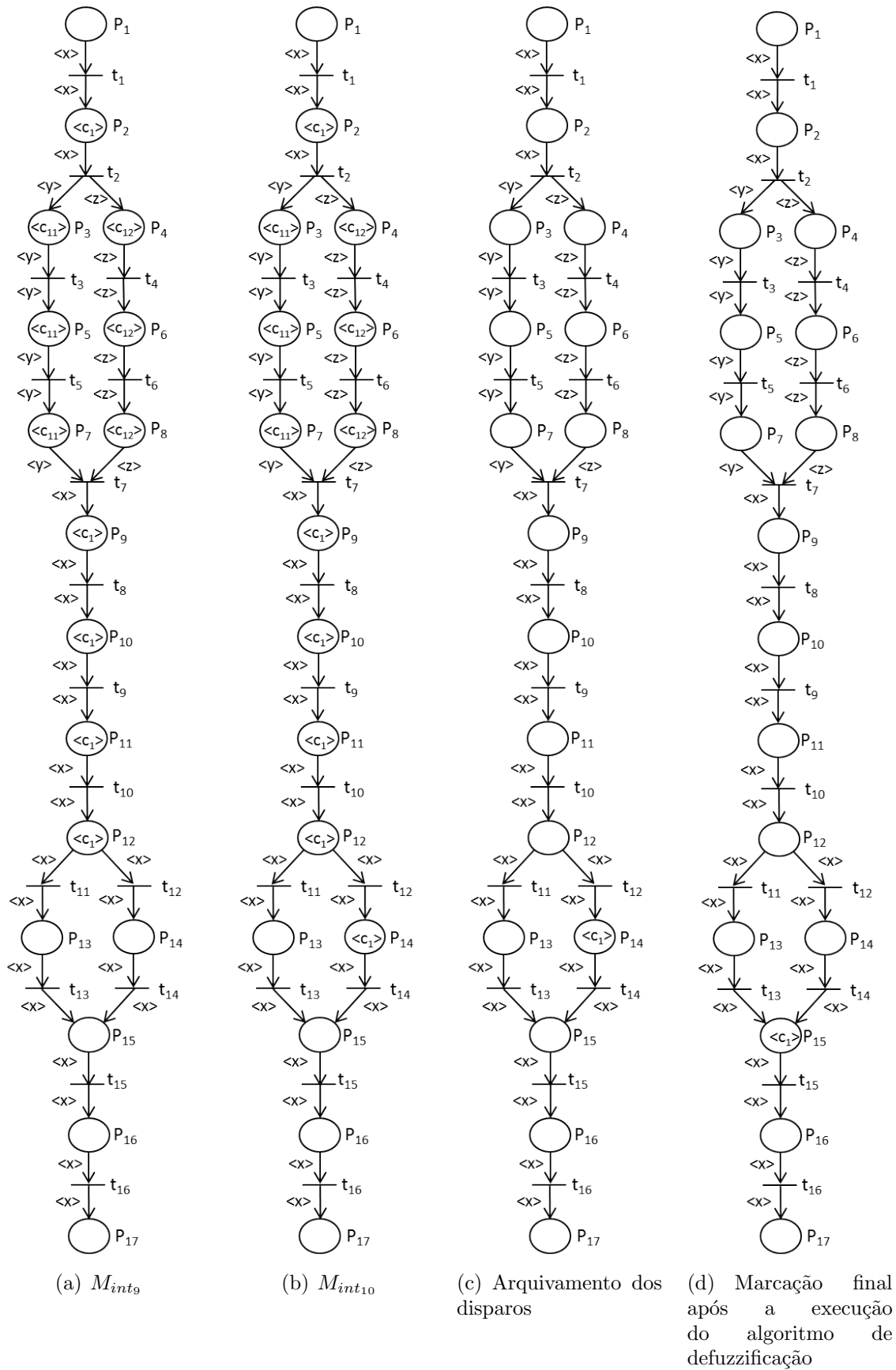


FIGURA 4.15: Resultados da simulação do cenário II.

P_{14} é 1 (um). Devido à marcação imprecisa do objeto $\langle c_1 \rangle$ em P_{14} , a transição t_{14} está habilitada e, dado que sua interpretação é verdadeira, o algoritmo de defuzzificação, detalhado na Seção 3.3.1, é chamado.

O algoritmo de defuzzificação tornará a marcação imprecisa da Figura 4.15(b) em precisa (Figura 4.15(c)), através do arquivamento dos disparos das transições $t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}$ e t_{12} (um novo cálculo da distribuição de possibilidade das cópias do objeto $\langle c_1 \rangle$, $\langle c_{11} \rangle$ e $\langle c_{12} \rangle$ é realizada a fim de voltar para a marcação precisa). Tradicionalmente, o algoritmo de defuzzificação tem a função de tornar certo o conhecimento incerto do estado do sistema quando alguns eventos ocorrem de forma inesperada.

Após a execução do algoritmo, a marcação global da *WorkFlow net* possibilística da Figura 4.15(c) torna-se precisa e consistente com a situação corrente do processo. Após arquivar o disparo das transições, o algoritmo realiza o disparo da transição t_{14} normalmente (Figura 4.15(d)), removendo o objeto $\langle c_1 \rangle$ do lugar P_{14} e produzindo um novo objeto em P_{15} , além de executar as ações associadas à transição t_{14} .

A sequência ordenada dos pseudo-disparos ($t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}$) obtida através da defuzzificação é registrada a fim de informar as atividades que deveriam ter sido executadas mas não foram quando se considera o modelo do processo.

Este cenário corresponde a uma situação em que o modelo não permite a execução de uma atividade sem que as suas anteriores não estejam devidamente executadas. Entretanto, através do uso da teoria das redes de Petri e dos disparos incertos, é possível ignorar diversas atividades anteriores, “saltando-as”.

Caso uma “rede de Petri clássica”, baseada no “jogador” de rede de Petri da Figura 4.1, seja considerada, tal situação poderá ser resolvida acrescentado uma transição t à Figura 4.11, a qual representa o caminho alternativo que permite “saltar” da atividade “Registrar” para a conclusão da atividade “Enviar Carta”. Essa modificação no modelo pode ser observada na Figura 4.16.

Pode-se observar que a transição t , quando disparada, produz um objeto no lugar P_{15} em vez de ser em P_{14} . Isto acontece, pois, ao receber o evento de conclusão do envio da carta, esta transição tem de comportar identicamente à transição t_{14} . Se o objeto fosse produzido em P_{14} um desvio ainda aconteceria, pois esta transição não estaria habilitada e o evento a tornaria verdadeira.

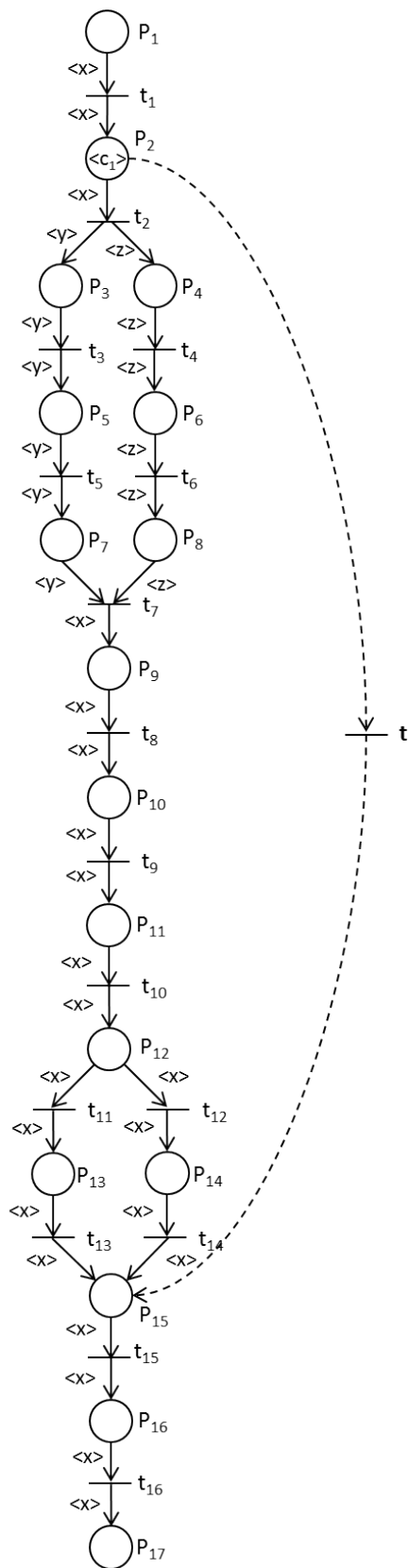


FIGURA 4.16: O processo “Serviço de Reclamações” modificado para suportar o cenário II.

4.3 Considerações Finais do Capítulo

Neste capítulo, foi apresentado definindo um tipo de “Workflow net possibilística”, a qual combina a estrutura de roteamento das “*WorkFlow nets*” com as marcações e disparos imprecisos das “redes de Petri possibilísticas”. Esta técnica permite uma certa flexibilidade de execução dos processos de negócios que geralmente são modelados com uma estrutura de controle complexa e rígida.

A proposta apresentada foca na flexibilidade por desvio na perspectiva do fluxo de controle [27, 28]. Tal flexibilidade caracteriza-se pela capacidade da instância do objeto desviar-se a partir das alternativas de execução descritas no modelo do processo sem modificá-lo.

Este modelo proposto é capaz de tolerar as intervenções e desvios humanos descrevendo situações onde o sistema não permite o prosseguimento do mesmo por não conhecer com certeza a sequência correta de ações à realizar. Logo, o número de estados acessíveis aumenta permitindo uma maior adequação do modelo do processo às ações realizadas pelos atores humanos.

A fim de exemplificar a abordagem, dois possíveis incidentes no processo “Serviço de Reclamação” (Figura 4.2) foram considerados. O primeiro incidente (Seção 4.1) ocorre por causa das restrições temporais que foram incluídas nas condições das transições do modelo e, o segundo (Seção 4.2), pela autonomia de decisão dada a um funcionário em função de seu conhecimento permitindo-o ignorar um conjunto completo de atividades especificadas e descritas no modelo do processo. Em ambos, foi necessário “saltar” uma ou mais atividades considerando o fluxo de controle definido no modelo.

O algoritmo de defuzzificação utilizado para restaurar o estado consistente do processo permite diagnosticar o que aconteceu de anormal na execução do mesmo através da sequência de disparo resultante da sua execução. A formalização de tal sequência informa o que deveria ter sido executado, mas não foi. Esta formalização não é obtida numa rede de Petri com várias alternativas, o que impede o conhecimento sobre quais atividades eram necessárias na execução do processo, mas não foram realizadas.

Caso uma “rede de Petri clássica”, baseada no “jogador” de rede de Petri da Figura 4.1, seja considerada diversas novas transições deveriam ser criadas a fim de considerar todos os cenários anormais possíveis. Como consequência, dificulta-se a legibilidade do modelo, o que pode ser observado na Figura 4.17 onde foram adicionadas algumas transições para exemplificar o aumento da complexidade do modelo.

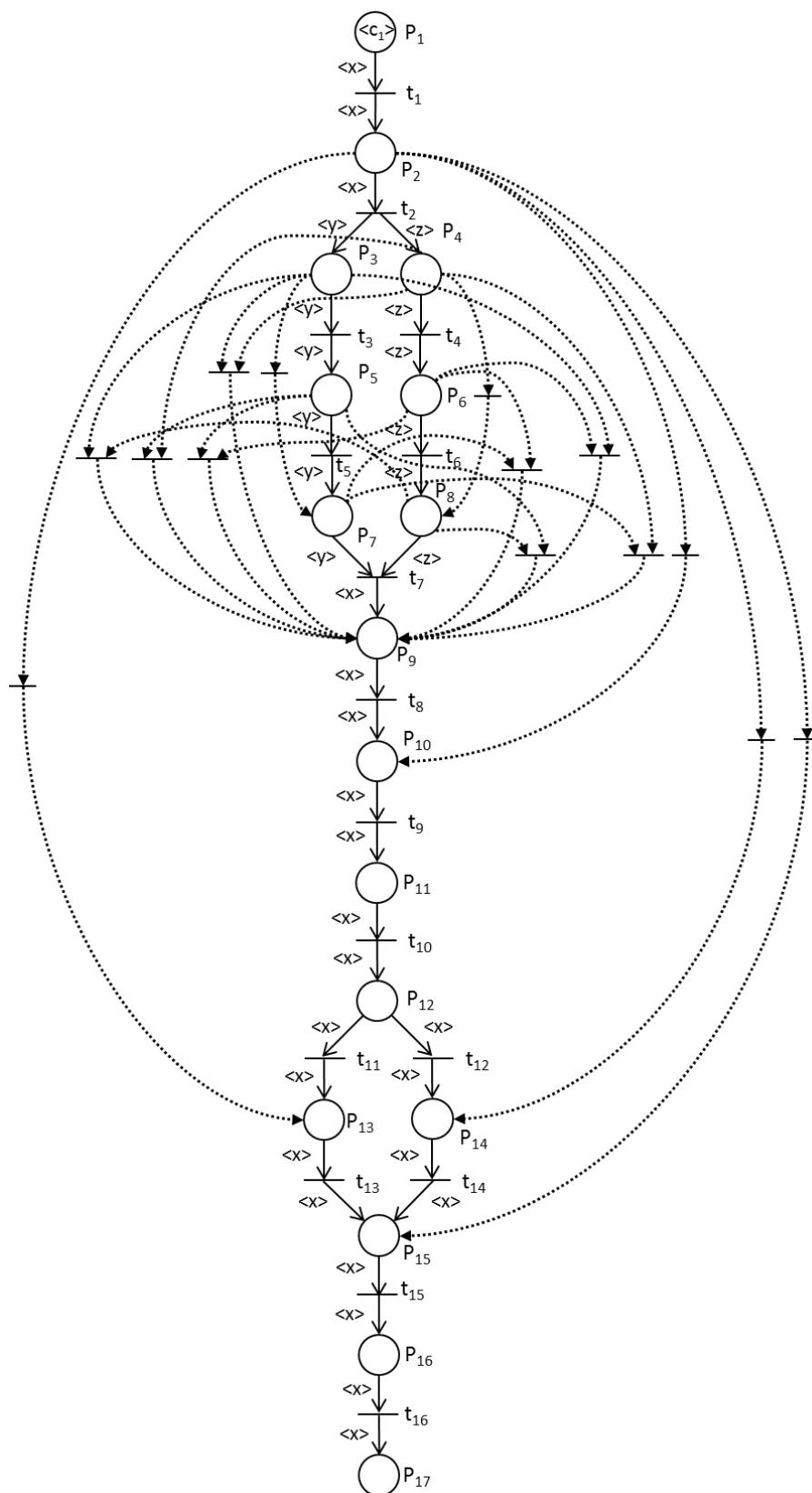


FIGURA 4.17: “Serviço de Reclamações” com a presença de novas transições.

Uma outra opção para o tratamento de tais incidentes seria reiniciar o processo manualmente. Consequentemente o processo ficaria temporariamente num estado de bloqueio impedindo o prosseguimento do mesmo além de gerar uma ociosidade no sistema.

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho foi apresentado o modelo de uma “WorkFlow net possibilística” que combina a estrutura de roteamento das WorkFlow Nets (redes de Petri que modelam processos de negócios) com a marcação e o disparo impreciso das “redes de Petri possibilística” definidas em [20]. Esta abordagem foca na flexibilidade por desvio na perspectiva do fluxo de controle [27, 28] sendo capaz de lidar com os possíveis desvios/inconsistências relacionados ao fluxo de controle do modelo que aparecem durante a execução em tempo real do processo quando atores humanos são envolvidos nas atividades do processo, recuperando o estado consistente do mesmo sem necessariamente alcançar estados inconsistentes.

O processo de um “Serviço de Reclamação”, apresentado em [14], onde atores humanos são envolvidos nas atividades do processo, juntamente com dois exemplos de desvios foram utilizados para ilustrar a abordagem proposta.

No primeiro, o ator responsável por entrar em contato com o cliente não consegue contatá-lo por causa das restrições temporais presentes nas condições incluídas nas transições do modelo. No segundo, a autonomia de decisão dada a um funcionário em função de seu conhecimento permitindo-o ignorar um conjunto completo de atividades representadas no modelo do processo faz com que uma atividade seja realizada sem que as suas precedentes sejam executadas.

Em ambos dos desvios é necessário “saltar” respectivamente uma ou mais atividades considerando o fluxo de controle definido no modelo, sem modificá-lo, para que o processo prossiga normalmente. Isso é realizado através da adoção do conceito de marcação e de disparo incerto. Caso não fosse utilizado tal artifício, o processo ficaria bloqueado até que o mesmo fosse reiniciado manualmente ou que cliente fosse contatado, no caso do primeiro desvio.

No segundo desvio, além do conceito de marcação e de disparo incerto, utiliza-se a equação fundamental para definir uma possível sequência de disparos. A ordem de execução desta sequência é dada pelo algoritmo de defuzzificação, o qual permite diagnosticar o que aconteceu de anormal na execução do processo. Numa rede de Petri com várias alternativas esta sequência não é obtida, o que impede o conhecimento sobre quais atividades eram necessárias na execução do processo, mas não foram realizadas, logo a formalização do desvio, neste modelo, não é trivial.

Comparando esta abordagem com outros trabalhos que tratam do problema de não conformidade, a principal vantagem é que a “rede de Petri clássica” (que modela um processo de workflow) não é modificada e, portanto os métodos clássicos para análise das “boas” propriedades permanecem válidos, tais como a propriedade *sound*. Além disso, a abordagem possibilística é adaptada para o conceito de flexibilidade e robustez no processo. Ademais, ela é capaz de “raciocinar” num conhecimento imperfeito sobre o estado do sistema, permitindo o prosseguimento do mesmo sem que possíveis bloqueios ocorram tornando assim mais robusto e mais “inteligente” a interação entre os atores humanos encarregados de executar as atividades e o modelo do processo.

Os disparos incertos (ou pseudo-disparos) das transições podem atualizar e revisar o conhecimento sobre o estado do sistema representado pela marcação e, essas noções são completamente consistentes com a teoria da rede de Petri, pois o conjunto real das marcações alcançáveis não são modificadas, logo as principais “boas” propriedades da WorkFlow net são preservadas.

Este trabalho teve como resultados, além da escrita desta dissertação, a publicação e apresentação dos artigos:

- “*Possibilistic WorkFlow nets to deal with non-conformance in Process Execution*” [62], aceito para publicação pela *IEEE International Conference on Systems, Man, and Cybernetics* (SMC 2012) realizado em Seul - Korea.
- “*Inconsistency recovery in Business Processes using a possibilistic WorkFlow net*” [63], aceito para publicação pela *XXXI International Conference of the Chilean Computer Science Society* (SCCC’2012) realizado em Valparaíso - Chile.

Como trabalhos futuros, seria interessante estender os resultados da abordagem proposta como por exemplo:

- considerar roteiros iterativos¹;

¹Estes roteiros representam as situações em que uma atividade pode ser executada mais de uma vez para um mesmo caso.

- contemplar de forma explícita os recursos humanos no modelo do processo como apresentado nos artigos [43, 54, 55, 64] ;
- trabalhar com marcações iniciais imprecisas e/ou com marcações finais parcialmente conhecidas numa situação onde uma condição de uma transição não habilitada se torna verdadeira. Esta situação não permite o uso da equação fundamental pelo fato das marcações serem parcialmente definidas;
- adaptar ou desenvolver um procedimento para recuperar os desvios que são gerados num processo não necessariamente *sound*, visto que, na prática, a flexibilidade inerente dos sistemas legados nem sempre permitem a produção de um modelo de processo que respeite tal propriedade.
- transformar a distribuição de possibilidade dos objetos num conjunto *fuzzy*, variando o valor da possibilidade num intervalo entre 0 (totalmente ausente) e 1 (totalmente presente) quando a localização do mesmo se considera incerta. A marcação *fuzzy* permite relaxar as restrições de sincronização que frequentemente são muito rigorosas e/ou produzir uma espécie de hierarquia entre um conjunto de possíveis alternativas transformando alguns eventos mais plausíveis do que outros. Essa abordagem deve ser interessante para a monitoração, a fim de diminuir a imprecisão da ação de intervenção humana.

Referências Bibliográficas

- [1] D. Hollingsworth. The workflow reference model. Technical report, Workflow Management Coalition, 1994.
- [2] W. M. P. van der Aalst. The application of petri nets to workflow management. *Journal of Circuits Systems and Computers*, 8(1):21 – 66, 1998, Citeseer.
- [3] W. M. P. van der Aalst, A. Hofstede, M. Adams, e N. Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer Publishing Company, Incorporated, 1st edition, 2009.
- [4] S. I. D. Pádua, A. R. Y. Silva, A. J. V. Porto, e R. Y. Inamasu. O potencial das redes de petri em modelagem e análise de processos de negócio. *Gestão & Produção*, 11(1):109 – 119, 2004.
- [5] R. Valette, M. Courvoisier, e D. Mayeux. Control of flexible production systems and petri nets. In Anastasia Pagnoni e Grzegorz Rozenberg, editors, *European Workshop on Applications and Theory of Petri Nets*, volume 66 of *Informatik-Fachberichte*, pp. 264 – 277. Springer, 1982.
- [6] J. Cardoso, R. Valette, e D. Dubois. Monitoring manufacturing systems by means of petri nets with imprecise markings. *IEEE International Symposium on Intelligent Control*, pp. 233 – 238, September 25 - 26 1989.
- [7] M. Silva e R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, volume 424 of *Lecture Notes in Computer Science*, pp. 274–417. Springer-Verlag, Berlin, 1990.
- [8] D. Andreu, J-C Pascal, e R. Valette. Fuzzy petri net-based programmable logic controller. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27:952 – 961, 1997.
- [9] J. Cardoso, R. Valette, e D. Dubois. Possibilistic petri nets. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(5):573 – 582, 1999.

- [10] O. L. Asato, F. Junqueira, D. J. dos Santos Filho, P. E. Miyagi, e L. O. Araujo. Control of productive systems with functional flexibility level. In *Emerging Technologies Factory Automation (ETFa)*, 2011 IEEE 16th Conference on, pp. 1 – 4, 2011.
- [11] O. L. Asato, G. M. Dobrianskyj, F. Junqueira, D. J. dos Santos Filho, e P. E. Miyagi. Process control system considering the machines functional flexibilities. In L. M. Camarinha-Matos, E. Shahamatnia, e G. Nunes, editors, *DoCEIS*, volume 372 of *IFIP Advances in Information and Communication Technology*, pp. 133 – 142. Springer, 2012.
- [12] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, e K. Schneider. Are developers complying with the process: an xp study. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pp. 14:1 – 14:10, New York, NY, USA, 2010. ACM.
- [13] S. Thompson e T. Torabi. An observational approach to practical process non-conformance detection. *Applications of Digital Information and Web Technologies 2009 ICADIWT 09 Second International Conference on the*, pp. 62 – 67, 2009.
- [14] W. M. P. van der Aalst. *Workflow Management: Models, Methods, and Systems*, volume 1 of *MIT Press Books*. The MIT Press, 1 edition, 2004.
- [15] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541 – 580, 1989.
- [16] S. Ling e H. Schmidt. Time petri nets for workflow modelling and analysis. *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, 4:3039 – 3044, 2000.
- [17] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965, Elsevier.
- [18] C. G. Looney. Fuzzy petri nets for rule-based decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18 (1):178 – 183, 1988.
- [19] T. Murata. Temporal uncertainty and fuzzy-timing high-level petri nets. In Jonathan Billington e Wolfgang Reisig, editors, *Application and Theory of Petri Nets 1996*, volume 1091 of *Lecture Notes in Computer Science*, pp. 11 – 28. Springer Berlin / Heidelberg, 1996.
- [20] J. Cardoso. *Time Fuzzy Petri Nets*, volume 22 of *Studies in Fuzziness and Soft Computing*, pp. 115 – 145. Physica-Verlag, 1999.

- [21] L. A. Zadeh. Fuzzy logic. *Computer*, 21:83 – 93, 1988, Los Alamitos, CA, USA, IEEE Computer Society.
- [22] D. Dubois e H. Prade. Possibility theory. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pp. 6927–6939. Springer, 2009.
- [23] G. L. Curry e R. M. Feldman. *Manufacturing Systems Modeling and Analysis*. Springer, segunda edition, 2011.
- [24] P. Heinl, S. Horn, S. Jablonski, J. Neeb, K. Stein, e M. Teschke. A comprehensive approach to flexibility in workflow management systems. *SIGSOFT Softw. Eng. Notes*, 24(10):79 – 88, March 1999, New York, NY, USA, ACM.
- [25] H. A. Reijers. Workflow flexibility: The forlorn promise. In *15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006)*, pp. 271 – 272. IEEE Computer Society, 2006.
- [26] R. A. Snowdon, B. Warboys, R. M. Greenwood, C. P. Holland, P. J. Kawalek, e D. R. Shaw. On the architecture and form of flexible process support. *Software Process: Improvement and Practice*, 12(1):21–34, 2007.
- [27] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, e W. M. P. van der Aalst. Process flexibility: A survey of contemporary approaches. In J. L. G. Dietz, A. Albani, e J. Barjis, editors, *4th International Workshop CIAO! and 4th International Workshop EOMAS*, volume 10 of *Lecture Notes in Business Information Processing*, pp. 16 – 30. Springer, 2008.
- [28] H. Schonenberg, R. Mans, N. Russell, N. Mulyar, e W. M. P. van der Aalst. Towards a taxonomy of process flexibility. In Zohra Bellahsene, Carson Woo, Ela Hunt, Xavier Franch, e Remi Coletta, editors, *Proceedings of the Forum at the CAiSE 08 Conference(CAiSE Forum)*, volume 344 of *CEUR Workshop Proceedings*, pp. 81 – 84. CEUR-WS.org, 2008.
- [29] R. David e H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer Publishing Company, Incorporated, 2nd edition, 2010.
- [30] W. M. P. van der Aalst e C. Stahl. *Modeling Business Processes: A Petri Net-Oriented Approach*. Cooperative Information Systems. Mit Press, 2011.
- [31] Y. T. Kotb e E. Badreddin. Synchronization among activities in a workflow using extended workflow petri nets. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, CEC '05*, pp. 548 – 551, Washington, DC, USA, 2005. IEEE Computer Society.

- [32] K. Mohammed, L. Redouane, e C. Bernard. A deviation-tolerant approach to software process evolution. In *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting, IWPSE '07*, pp. 75 – 78, New York, NY, USA, 2007. ACM.
- [33] G. Cugola, E. Di Nitto, C. Ghezzi, e M. Mantione. How to deal with deviations during process model enactment. In *Proceedings of the 17th International Conference on Software Engineering*, pp. 265 – 265, April 1995.
- [34] G. Cugola, E. Di Nitto, A. Fuggetta, e C. Ghezzi. A framework for formalizing inconsistencies and deviations in human-centered systems. *ACM Trans. Softw. Eng. Methodol.*, 5(3):191–230, July 1996, New York, NY, USA, ACM.
- [35] A. Rozinat e W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64 – 95, 2008.
- [36] W. M. P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, e H. M. W. Verbeek. Choreography conformance checking: An approach based on bpel and petri nets. December 2005.
- [37] J. Munoz-Gama. Algorithms for process conformance and process refinement. Master's thesis, Universitat Politècnica de Catalunya (UPC), sep 2010.
- [38] A. Adriansyah, B. F. van Dongen, e W. M. P. van der Aalst. Towards robust conformance checking. In *Business Process Management Workshops'10*, pp. 122 – 133, 2010.
- [39] J. Munoz-Gama e J. Carmona. A fresh look at precision in process conformance. In Richard Hull, Jan Mendling, e Stefan Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pp. 211 – 226. Springer, 2010.
- [40] W. M. P. van der Aalst, A. Adriansyah, e B. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182 – 192, 2012, John Wiley & Sons, Inc.
- [41] M. A. A. da Silva, R. Bendraou, X. Blanc, e M.-P. Gervais. Early deviation detection in modeling activities of mde processes. In *Proceedings of the 13th international conference on Model driven engineering languages and systems: Part II, MODELS'10*, pp. 303 – 317, Berlin, Heidelberg, 2010. Springer-Verlag.

- [42] T. Murata, T. Suzuki, e S. Shatz. *Fuzzy-Timing High-Level Petri Nets (FTHNs) for Time-Critical Systems*, volume 22 of *Studies in Fuzziness and Soft Computing*, pp. 88 – 114. Physica-Verlag, 1999.
- [43] J. C. Jeske, S. Julia, e R. Valette. Fuzzy continuous resource allocation mechanisms in workflow management systems. *Software Engineering, Brazilian Symposium on*, 0:236 – 251, 2009, Los Alamitos, CA, USA, IEEE Computer Society.
- [44] S. Cîmpan e F. Oquendo. Dealing with software process deviations using fuzzy logic based monitoring. *SIGAPP Appl. Comput. Rev.*, 8(2):3 – 13, December 2000, New York, NY, USA, ACM.
- [45] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962.
- [46] J. Cardoso e R. Valette. *Redes de Petri*. DAUSFC, 1997.
- [47] M. Diaz. *Petri Nets: Fundamental Models, Verification and Applications*. Wiley-IEEE Press, 2009.
- [48] C. Sibertin-Blanc. High level petri nets with data structure. *6th European Workshop in Application and Theory of Petri Nets.*, pp. 141 – 170, June 1985.
- [49] C. Sibertin-Blanc. Cooperative objects: Principles, use and implementation. In G. Agha, F. De Cindio, e G. Rozenberg, editors, *Concurrent Object-Oriented Programming and Petri Nets*, volume 2001 of *Lecture Notes in Computer Science*, pp. 216 – 246. Springer Berlin / Heidelberg, 2001.
- [50] J. Cardoso, R. Valette, e D. Dubois. Petri nets with uncertain markings. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pp. 64 – 78. Springer Berlin / Heidelberg, 1991.
- [51] R. Valette. Analysis of petri nets by stepwise refinements. *Journal of Computer and System Sciences*, 18(1):35 – 46, 1979.
- [52] W. M. P. van der Aalst. Modeling and analyzing interorganizational workflows. In L. Lavagno e W. Reisig, editors, *Proceedings of the 1998 International Conference on Application of Concurrency to System Design (CSD'98)*, pp. 262 – 272, Fukushima, Japan, March 1998. IEEE Computer Society Press.
- [53] L. M. Soares Passos e S. Julia. Qualitative analysis of workflow nets using linear logic: Soundness verification. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 2843 – 2847, 2009.

- [54] S. Julia, F. F. de Oliveira, e R. Valette. Real time scheduling of workflow management systems based on a p-time petri net model with hybrid resources. *Simulation Modelling Practice and Theory*, 16(4):462–482, 2008.
- [55] J. C. Jeske, S. Julia, e R. Valette. A model to represent human behavior in workflow management systems. *ICNPAA 2010 World Congress: 8th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences*, pp. 385 – 392, 2010, São José dos Campos, Brazil.
- [56] A. Rozinat, I. S. M. De Jong, C. W. Günther, e W. M. P. van der Aalst. Conformance analysis of asml’s test process. *Proceedings of the Second International Workshop on Governance, Risk and Compliance (GRCIS’09)*, 459:1 – 15, 2009, CEUR-WS.org.
- [57] M. Pesic. *Constraint-Based Workflow Management Systems: Shifting Control to Users*. PhD thesis, Eindhoven University of Technology, 2008.
- [58] W. M. P. van der Aalst, K. M. van Hee, J. M. van der Werf, e M. Verdonk. Auditing 2.0: Using process mining to support tomorrow’s auditor. *Computer*, 43(3):90 – 93, March 2010, Los Alamitos, CA, USA, IEEE Computer Society Press.
- [59] K. Barkaoui e I. B. Abdallah. Deadlock avoidance in fms based on structural theory of petri nets. In *Emerging Technologies and Factory Automation, 1995. ETFA ’95, Proceedings., 1995 INRIA/IEEE Symposium on*, volume 2, pp. 499 – 510, 1995.
- [60] K. Barkaoui e J.-F. Pradat-Peyre. On liveness and controlled siphons in petri nets. In Jonathan Billington e Wolfgang Reisig, editors, *Proceedings of the 17th International Conference on Application and Theory of Petri Nets*, volume 1091 of *Lecture Notes in Computer Science*, pp. 57 – 72, London, UK, UK, 1996. Springer-Verlag.
- [61] E. R. Boer e T. Murata. Generating basis siphons and traps of petri nets using the sign incidence matrix. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 41:266 – 271, 1994.
- [62] L. P. Rezende, S. Julia, e J. Cardoso. Possibilistic workflow nets to deal with non-conformance in process execution. In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, Seoul, Korea, 2012.
- [63] L. P. Rezende, S. Julia, e J. Cardoso. Inconsistency recovery in business processes using a possibilistic workflow net. In *Proceedings of the 2012 International Conference of the Chilean Computer Science Society*, Valparaíso, Chile, 2012.

- [64] S. Julia e F. F. Oliveira. A p-time hybrid petri net model for the scheduling problem of workflow management systems. In *SMC (5)'04*, pp. 4947 – 4952, 2004.