

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



**FORMALIZAÇÃO DE WORKFLOW NETS UTILIZANDO
LÓGICA LINEAR: ANÁLISE QUALITATIVA E
QUANTITATIVA**

LÍGIA MARIA SOARES PASSOS

Uberlândia - Minas Gerais

2009

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



LÍGIA MARIA SOARES PASSOS

FORMALIZAÇÃO DE WORKFLOW NETS UTILIZANDO LÓGICA LINEAR: ANÁLISE QUALITATIVA E QUANTITATIVA

Dissertação de Mestrado apresentada à Faculdade de Ciência da 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

Co-orientador:

Prof. Dr. Marcelo de Almeida Maia

Uberlândia, Minas Gerais

2009

Dados Internacionais de Catalogação na Publicação (CIP)

- P289f Passos, Lúgia Maria Soares, 1984-
 Formalização de workflow nets utilizando lógica linear: análise qualitativa e quantitativa / Lúgia Maria Soares Passos. - 2009.
 102 f. : il.
- Orientador: Stéphane Julia.
 Co-orientador: Marcelo de Almeida Maia.
 Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação.
 Inclui bibliografia.
1. Engenharia de software - Teses. 2. Fluxo de trabalho - Teses. I. Julia, Stéphane. II. Maia, Marcelo de Almeida. III. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU: 681.3.06

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

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

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Ciência da Computação a aceitação da dissertação intitulada “**Formalização de WorkFlow nets utilizando Lógica Linear: Análise Qualitativa e Quantitativa**” por **Lígia Maria Soares Passos** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 27 de Maio de 2009

Orientador:

Prof. Dr. Stéphane Julia
Universidade Federal de Uberlândia

Co-orientador:

Prof. Dr. Marcelo de Almeida Maia
Universidade Federal de Uberlândia

Banca Examinadora:

Prof. Dr. Carlos Roberto Lopes
Universidade Federal de Uberlândia

Profa. Dra. Emilia Villani
Instituto de Tecnologia Aeronáutica

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

Data: Maio de 2009

Autor: **Lígia Maria Soares Passos**
Título: **Formalização de WorkFlow nets utilizando Lógica Linear:
Análise Qualitativa e Quantitativa**
Faculdade: **Faculdade de Ciência da 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.

Dedicatória

*Dedico este trabalho aos meus pais Milton e Cidinha,
às minhas irmãs Aline e Giselle, ao meu irmão Leonardo e
ao meu namorado Luís Fernando.*

Agradecimentos

Gostaria de agradecer...

À Deus, pela dádiva da vida, pela oportunidade dessa existência, pela proteção, força e amparo contínuos.

Ao meu anjo guardião e amigos espirituais, por toda luz e companhia constante.

Aos meus pais Juvenal Milton Passos e Maria Aparecida Soares Passos pelo apoio, confiança, atenção, paciência, carinho e amor em todos os momentos dessa minha vida.

Às minhas irmãs Aline Soares Passos e Giselle Soares Passos pelos conselhos, incentivo, paciência e amor, em todas as fases (mesmo nas difíceis)...

Ao meu irmão Leonardo Brás Soares Passos, pela extrema paciência, carinho, cuidado e amor, sempre demonstrados.

Ao meu eterno namorado Luís Fernando Sant'Ana Borsari pela paciência, amizade, carinho, companheirismo e, principalmente, pelo amor incondicional que desde o início me sustenta.

Aos meus cunhados Marcos Gonçalves de Santana e Sérgio Ferreira Neves pela amizade, carinho e por todos os conselhos durante as viagens entre Franca e Uberlândia.

Aos meus avós, minhas tias, tios, primas e primos por sempre terem torcido por mim e por terem me apoiado em minhas decisões por mais que estas lhes parecem incoerentes.

Aos meus dois grandes e eternos amigos Tarcísio Abadio de Magalhães Júnior e Robson de Carvalho Soares, por todos os trabalhos feitos em conjunto, por todos os dias e noites de estudo, pelos conselhos, pela amizade verdadeira e carinho de sempre!

Aos meus amigos Jean Carlo de Souza Santos, Raquel Fialho de Queiroz Lafetá, Danilo e Camila Bettoni Molina, Thiago Resende Bisco, Daniel Vincenzi Romualdo da Silva e Rodrigo César Rocha dos Santos, pela amizade e carinho, mesmo quando distantes fisicamente.

À professora Márcia Aparecida Fernandes, pela força, incentivo, carinho e amizade.

À professora Sandra Aparecida de Amo, por todo o apoio, mesmo antes do ingresso na graduação.

Ao professor Marcelo de Almeida Maia, pela co-orientação desta pesquisa.

À professora Rita Maria da Silva Julia, pelo carinho sempre demonstrado.

Em especial ao professor Stéphane Julia pelo profissionalismo, apoio, paciência, confiança, amizade e orientação em relação à vida e à esta pesquisa.

Aos funcionários da secretaria do Programa de Pós-Graduação em Ciência da Computação da UFU, Maria Helena e Erisvaldo.

À CAPES, pelo apoio financeiro.

Resumo

Este trabalho apresenta um método para a análise qualitativa e quantitativa de Workflow nets baseado nas árvores de prova canônica da lógica linear e uma abordagem para a verificação de especificações de processos de workflow em UML através da transformação de Diagramas de Atividades da UML em Workflow nets.

A análise qualitativa refere-se à prova do critério de corretude *soundness* definido para Workflow nets.

Já a análise quantitativa preocupa-se com o planejamento de recursos para cada atividade de um processo de workflow mapeado em uma t-Time Workflow net e baseia-se no cálculo de datas simbólicas para o planejamento de recursos utilizados na realização de cada tarefa do processo de workflow.

Para a verificação das especificações de processos de workflow mapeados em Diagramas de Atividades da UML são apresentadas regras formais para transformar estes diagramas em Workflow nets. Neste contexto também é proposta a análise e correção de pontos críticos em Diagramas de Atividades da UML através da análise de árvores de prova canônica da lógica linear.

As vantagens das abordagens apresentadas neste trabalho são diversas. O fato de trabalhar com lógica linear permite provar o critério de corretude *soundness* em tempo linear e sem que seja necessária a construção de um grafo das marcações acessíveis, considerando diretamente a própria estrutura da Workflow net, ao invés de considerar o seu autômato correspondente.

Além disso, o cálculo de datas simbólicas correspondentes à execução de cada tarefa mapeada em uma t-Time Workflow net permite planejar a utilização dos recursos envolvidos nas atividades do processo de workflow, através de fórmulas que podem ser utilizadas por qualquer caso tratado pelo processo de workflow correspondente, sem que seja necessário percorrer novamente o processo de workflow inteiro para recalcular, para cada novo caso, datas de início e término das atividades envolvidas no processo.

Já no que diz respeito à verificação de processos de workflow mapeados em Diagramas de Atividades da UML, a principal vantagem desta abordagem é a transformação de um modelo semi-formal em um modelo formal, para o qual algumas propriedades, como *soundness*, podem ser formalmente verificadas.

Palavras chave: processos de workflow, workflow nets, soundness, lógica linear, planejamento de recursos, diagrama de atividades da uml, atl.

Abstract

This work presents a method for qualitative and quantitative analysis of WorkFlow nets based on the proof trees of linear logic, and an approach for the verification of workflow specifications in UML through the transformation of UML Activity Diagrams into WorkFlow nets.

The qualitative analysis is concerned with the proof of soundness correctness criterion defined for WorkFlow nets.

The quantitative analysis is based on the computation of symbolic dates for the planning of resources used to handle each task of the workflow process modeled by a t-Time WorkFlow net.

For the verification of the specifications of workflow processes mapped into UML Activity Diagrams are presented formal rules to transform this ones into WorkFlow nets. In this context is proposed the analysis and correction of critical points in UML Activity Diagrams through the analysis of proof trees of linear logic.

The advantages of such an approach are diverse. The fact of working with linear logic permits one to prove the correctness criterion soundness in a linear time without considering the construction of the reachability graph, considering the proper structure of the WorkFlow net instead of considering the corresponding automata.

Moreover, the computation of symbolic dates for the execution of each task mapped into the t-Time WorkFlow net permits to plan the utilization of the resources involved in the activities of the workflow process, through formulas that can be used for any case handled by the correspondent workflow process, without to examine again the process to recalculate, for each new case, the dates of start and conclusion for the activities involved in the process.

Regarding the verification of workflow processes mapped into UML Activity Diagrams, the major advantage of this approach is the transformation of a semi-formal model into a formal model, such that some properties, like soundness, can be formally verified.

Keywords: workflow process, workflow nets, soundness, linear logic, resource planning, uml activity diagrams, atl.

Sumário

Lista de Figuras	xvii
Lista de Tabelas	xix
Lista de Abreviaturas e Siglas	xxi
1 Introdução	23
1.1 Contribuições	24
1.2 Organização da dissertação	25
2 Redes de Petri e WorkFlow nets	27
2.1 Redes de Petri	27
2.2 WorkFlow nets	28
2.2.1 Processos	29
2.2.2 Roteamentos	30
2.2.3 Acionamentos	31
2.2.4 Metamodelo das WorkFlow nets	31
2.3 <i>Soundness</i>	32
2.4 t-Time WorkFlow nets	33
2.5 Considerações Finais do Capítulo	34
3 Redes de Petri e Lógica Linear	35
3.1 Lógica Linear	35
3.1.1 Conectivos da Lógica Linear	36
3.2 Representação de Redes de Petri usando Lógica Linear	36
3.3 Cálculo de Sequentes	37
3.4 Árvores de Prova Canônica com Cálculo de Datas	39
3.5 Considerações Finais do Capítulo	41
4 ATL e Diagramas de Atividades da UML	43
4.1 ATL - <i>Atlas Transformation Language</i>	43
4.2 Diagramas de Atividades da UML	44

4.2.1	Metamodelo dos Diagramas de Atividades	45
4.3	Considerações Finais do Capítulo	47
5	Análise Qualitativa e Quantitativa de Workflow nets	49
5.1	Análise Qualitativa: Verificação da Propriedade <i>Soundness</i>	49
5.2	Análise Quantitativa: Planejamento de Recursos	54
5.3	Considerações Finais do Capítulo	58
6	Verificação de Especificações de Workflow em UML usando Transformações Automáticas para Workflow nets	59
6.1	Transformações Automáticas de Diagramas de Atividades da UML em Workflow nets	59
6.1.1	Regras para a transformação de um Diagrama de Atividades da UML em uma Workflow net	60
6.2	Encontrando e Corrigindo Pontos Críticos em Diagramas de Atividades da UML utilizando Árvore de Prova Canônica da Lógica Linear	65
6.3	Considerações Finais do Capítulo	70
7	Estudo de Caso	71
7.1	Análise Qualitativa e Quantitativa de uma Iteração do RUP	71
7.1.1	RUP - <i>Rational Unified Process</i>	71
7.1.2	Transformação do Diagrama de Atividades da UML que Modela uma Iteração do RUP em uma Workflow net	74
7.1.3	Análise Qualitativa: Verificação da Propriedade <i>Soundness</i> para uma Iteração do RUP	75
7.1.4	Análise Quantitativa: Planejamento de Recursos para uma Iteração do RUP	81
7.2	Considerações Finais do Capítulo	92
8	Conclusão e Trabalhos Futuros	93
	Referências Bibliográficas	95

Lista de Figuras

2.1	Exemplos de sensibilização e disparo de transição em uma rede de Petri.	28
2.2	Elementos de Modelagem de uma WorkFlow net.	29
2.3	WorkFlow net para o processo de tratamento de reclamações e os seus acionamentos.	29
2.4	Bloco bem formado.	30
2.5	Metamodelo das WorkFlow nets.	32
3.1	Redes de Petri para a exemplificação da tradução de redes de Petri em fórmulas da lógica linear.	37
3.2	Rede de Petri para exemplificação da construção de uma árvore de prova canônica da lógica linear.	39
3.3	Rede de Petri t-temporal para exemplificação da construção de uma árvore de prova canônica da lógica linear com cálculo de datas.	40
4.1	Exemplo de transformação em ATL.	44
4.2	Elementos de Modelagem de um Diagrama de Atividades da UML.	45
4.3	Metamodelo dos Diagramas de Atividades da UML.	46
4.4	Exemplo de Diagrama de Atividades da UML com seu respectivo Diagrama de Objetos. (a) Diagrama de Atividades. (b) Diagrama de Objetos.	46
5.1	WorkFlow net, com erro de modelagem, para o processo de tratamento de reclamações.	53
5.2	t-Time WorkFlow net com datas simbólicas para o processo de tratamento de reclamação e seus acionamentos.	55
5.3	t-Time WorkFlow net com datas numéricas para o processo de tratamento de reclamação e seus acionamentos.	57
6.1	Regras de transformação.	61
6.2	Exemplo de transformação.	64
6.3	O processo de tratamento de reclamações. (a) Processo de workflow modelado através de um Diagrama de Atividades da UML. (b) Processo de workflow modelado através de uma WorkFlow net.	64

6.4	Diagrama de Atividades da UML que modela o processo de seleção de peças.	66
6.5	WorkFlow net que modela o processo de seleção de peças.	66
6.6	Diagrama de Atividades da UML que modela o processo de seleção de peças corrigido.	69
6.7	WorkFlow net que modela o processo de seleção de peças corrigido.	70
7.1	Arquitetura global do RUP.	72
7.2	Diagrama de Atividades da UML que modela uma iteração do RUP.	73
7.3	WorkFlow net que modela uma iteração do RUP.	76
7.4	t-Time WorkFlow net, com datas simbólicas, que modela uma iteração do RUP.	87
7.5	t-Time WorkFlow net, com datas numéricas, que modela uma iteração do RUP.	88

Lista de Tabelas

3.1	Datas simbólicas de produção e consumo dos átomos da rede de Petri t-temporal da Figura 3.3.	41
3.2	Intervalos de datas simbólicas de produção e consumo dos átomos da rede de Petri t-temporal da Figura 3.3.	41
5.1	Intervalos de datas simbólicas para execução de tarefas do tipo usuário dos cenários Sc_1 e Sc_2	58
5.2	Intervalos de datas numéricas para a execução de tarefas do tipo usuário dos cenários Sc_1 e Sc_2	58
7.1	Datas de início ao mais cedo para cada um dos fluxos de engenharia do processo RUP, considerando o cenário Sc_2	91
7.2	Datas de término ao mais tarde para cada um dos fluxos de engenharia do processo RUP, considerando o cenário Sc_2	92

Lista de Abreviaturas e Siglas

ATL	Atlas Transformation Language
OCL	Object Constraint Language
OMG	Object Management Group
RUP	Rational Unified Process
UML	Unified Modeling Language

Capítulo 1

Introdução

O principal objetivo dos Sistemas de Gerenciamento de Workflow é executar processos de workflow. Os processos de workflow representam as sequências de atividades que devem ser executadas em uma organização para tratar casos específicos e alcançar uma meta bem definida [van der Aalst e van Hee 2004].

De acordo com [Murata 1989], as redes de Petri são apropriadas para modelar Sistemas de Tempo Real, uma vez que elas permitem uma boa representação de situações de conflito, compartilhamento de recursos, comunicação síncrona e assíncrona, restrições de precedência e restrições de tempo explícito, no caso das redes de Petri temporizadas.

Muitos trabalhos já consideraram a teoria das redes de Petri como uma ferramenta eficiente na modelagem e análise de Sistemas de Gerenciamento de Workflow. Em [van der Aalst 1998] e em [van der Aalst e van Hee 2004], por exemplo, são definidas as Workflow nets, que são redes de Petri que modelam processos de workflow. Em [Kotb e Badreddin 2005], é definido um modelo estendido de redes de Petri para a modelagem de workflow. Tal modelo permite o tratamento de recursos críticos que devem ser utilizados em atividades específicas em tempo real. Em [Ling e Schmidt 2000], 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. Em particular, um exemplo de sistema de saúde é usado para ilustrar uma abordagem onde uma enfermeira que cuida de dois pacientes é representada por uma ficha simples em um lugar compartilhado. Em [Vilallonga et al. 2003], especificações de “clock” são combinadas com a teoria das redes de Petri para a especificação de requisitos temporais em processos de negócio. Esta nova formalização permite algumas verificações temporais que dependem da construção de um grafo das marcações acessíveis. Em [Lin e Qu 2004], a análise quantitativa proposta é baseada em reduções de padrões básicos de workflow. Reduções sucessivas levam à perda de informação em relação ao tempo de execução de cada tarefa, individualmente. Assim, é possível responder quando o processo global finalizará. No entanto, não é possível responder questões sobre quando cada tarefa será executada e, conseqüentemente, planejar a utilização de recursos envolvidos nestas.

Há várias técnicas para analisar processos de workflow. [van der Aalst e van Hee 2004] definem duas metodologias de análise distintas: *análise qualitativa* e *análise quantitativa*. A análise qualitativa preocupa-se com a corretude de uma Workflow net, enquanto a análise quantitativa preocupa-se com a performance e dimensionamento de recursos.

Segundo [van der Aalst e van Hee 2004], o problema de planejamento de recursos tem como meta mostrar quais recursos e de que tipo são necessários considerando um dado período de tempo. Em particular, é importante encontrar um equilíbrio entre os recursos requisitados e os disponíveis.

Neste trabalho, uma abordagem baseada na lógica linear é proposta para analisar qualitativamente e quantitativamente uma Workflow net. A análise qualitativa refere-se à prova do critério de corretude *soundness* definido para Workflow nets [van der Aalst 1998]. Já a análise quantitativa preocupa-se com o planejamento de recursos para cada atividade de um processo mapeado em uma t-Time Workflow net.

Além disso, uma abordagem, baseada na linguagem de transformação ATL (*Atlas Transformation Language*), é proposta para a verificação de especificações de processos de workflow mapeados em Diagramas de Atividades da UML através de transformações automáticas para Workflow nets. Neste contexto é apresentada a formalização das regras para a transformação de um Diagrama de Atividades da UML em uma Workflow net e é proposta a análise e correção de pontos críticos em Diagramas de Atividades da UML através da análise de árvores de prova canônica da lógica linear.

1.1 Contribuições

Este trabalho apresenta quatro contribuições principais:

- A análise qualitativa de Workflow nets, que provê a prova do critério de corretude *soundness*, definido para estas redes, em tempo linear e mantendo a própria estrutura da rede, sem que seja necessária a construção de um autômato correspondente, através da análise de árvores de prova canônica da lógica linear.
- A análise quantitativa de Workflow nets, que provê o cálculo de intervalos de datas simbólicas para a execução de cada atividade presente em um processo de workflow mapeado através de uma t-Time Workflow net, possibilitando a reutilização destes intervalos de datas sem que seja necessário recalculá-los para cada novo caso (instância) do processo de workflow.
- A verificação de especificações de processos de workflow mapeados em Diagramas de Atividades da UML usando transformações automáticas para Workflow nets, que provêem:
 - A formalização das regras para a transformação de um Diagrama de Atividades da UML em uma Workflow net, ou seja, a formalização da transformação de

- um modelo semi-formal em um modelo formal, para o qual boas propriedades podem ser formalmente verificadas.
- A análise e correção de pontos críticos em Diagramas de Atividades da UML através da análise das árvores de prova canônica da lógica linear, construídas para a verificação da propriedade *soundness* para a WorkFlow net correspondente ao Diagrama de Atividades da UML.
 - A formalização de uma iteração do processo de desenvolvimento de software denominado RUP (*Rational Unified Process*) através da transformação de um Diagrama de Atividades da UML, que modela as atividades do fluxo de engenharia definido para este processo de desenvolvimento, em uma WorkFlow net, seguida da prova da propriedade *soundness* para esta WorkFlow net e do cálculo de intervalos de datas simbólicas para a realização de cada atividade presente neste processo.

1.2 Organização da dissertação

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

Nos Capítulos 2, 3 e 4 são apresentadas as fundamentações teóricas necessárias para o entendimento deste trabalho, sendo organizados como se segue.

O Capítulo 2 apresenta as redes de Petri e as WorkFlow nets, sendo que a Seção 2.1 faz uma breve introdução sobre o que são as redes de Petri, define seus elementos e as regras de sensibilização e disparo de uma transição. A Seção 2.2 define as WorkFlow nets, apresentando seus elementos de modelagem, a definição de processos, roteamentos e acionamentos e o metamodelo para estas redes. Já Seção 2.3 apresenta o critério de corretude *soundness*, definido para as WorkFlow nets. Finalmente, a Seção 2.4 define as t-Time WorkFlow nets, que serão utilizadas na análise quantitativa proposta neste trabalho.

O Capítulo 3 apresenta a lógica linear e suas relações com as redes de Petri. A Seção 3.1 apresenta a lógica linear e os seus conectivos. A Seção 3.2 mostra a representação das redes de Petri usando a lógica linear. A Seção 3.3 apresenta o cálculo de sequentes da lógica linear e suas regras. Por fim, as árvores de prova canônica da lógica linear com cálculo de datas são apresentadas na Seção 3.4.

No Capítulo 4, é apresentada a linguagem de transformações ATL e os Diagramas de Atividades da UML. A Seção 4.1 apresenta a linguagem de transformação ATL. Os Diagramas de Atividades da UML, assim como o seu metamodelo, são apresentados na Seção 4.2.

No Capítulo 5 são apresentadas abordagens para a análise qualitativa e quantitativa de WorkFlow nets. A análise qualitativa é apresentada na Seção 5.1. Já a Seção 5.2 apresenta a análise quantitativa.

No Capítulo 6 é apresentada a abordagem para a verificação de especificações de processos de workflow usando UML através de transformações automáticas para WorkFlow nets. A Seção 6.1 apresenta a transformação de Diagramas de Atividades da UML em WorkFlow nets. A Seção 6.2 mostra como encontrar e corrigir pontos críticos em Diagramas de Atividades da UML através da análise de árvores de prova canônica da lógica linear, sendo que para isso é tratado um exemplo.

O Capítulo 7 apresenta um estudo de caso onde é tratada uma iteração do processo de desenvolvimento RUP (*Rational Unified Process*). Assim, a Seção 7.1 trata da análise qualitativa e quantitativa de uma iteração do RUP e apresenta brevemente o processo de desenvolvimento RUP, descreve seus fluxos de engenharia e de suporte, apresenta a transformação de um Diagrama de Atividades da UML onde aparecem os fluxos de engenharia do RUP em uma WorkFlow net, verifica a propriedade *soundness* para a WorkFlow net gerada, apresenta a análise quantitativa desta WorkFlow net, sendo calculados os intervalos de datas simbólicas para a realização de cada atividade presente no fluxo de engenharia do processo RUP.

Finalmente, o Capítulo 8 apresenta a conclusão deste trabalho e as perspectivas de trabalhos futuros.

Capítulo 2

Redes de Petri e WorkFlow nets

Este capítulo tem por objetivo apresentar os conceitos relacionados às redes de Petri e às WorkFlow nets necessários para o entendimento deste trabalho. Para tanto, a Seção 2.1 define o que é uma rede de Petri. Já a Seção 2.2 apresenta as WorkFlow nets, a definição de processos, roteamentos, acionamentos e o metamodelo das WorkFlow nets. O critério de corretude das WorkFlow nets, denominado *soundness*, é definido na Seção 2.3. Finalmente, a Seção 2.4 apresenta as t-Time WorkFlow nets.

2.1 Redes de Petri

A teoria inicial das redes de Petri foi proposta em [Petri 1962]. Uma rede de Petri [Valette e Cardoso 1997], [Murata 1989] é uma ferramenta gráfica e um modelo formal abstrato que pode ser utilizada para a modelagem de diversos tipos de sistemas. O fato de ser formal permite, entre outras características, a verificação formal de boas propriedades.

Definição 2.1. (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 se, e somente se, existe um arco direcionado de p para uma transição t . Por exemplo, considerando a rede de Petri da Figura 2.1(a), os lugares $P1$ e $P2$ são lugares de entrada da transição t_1 .

Um lugar p é chamado de lugar de saída se, e somente se, existe um arco direcionado de t para p . Por exemplo, considerando a rede de Petri da Figura 2.1(a), o lugar $P3$ é um lugar de saída da transição t_1 .

Um lugar p contém em um dado momento zero ou mais fichas (tokens), que são representadas por pontos pretos. Por exemplo, considerando a rede de Petri da Figura 2.1(a), o lugar $P1$ contém uma ficha e o lugar $P2$ contém zero fichas.

A marcação de uma rede de Petri 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: 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 $t1$ da rede de Petri da Figura 2.1(a) não está sensibilizada. Já a transição $t1$ da rede de Petri da Figura 2.1 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 2.1(b) e 2.1(c) mostram um exemplo de disparo de transição. Neste caso, $t1$ consome uma ficha de cada lugar de entrada ($P1$ e $P2$) e produz uma ficha em cada lugar de saída ($P3$).

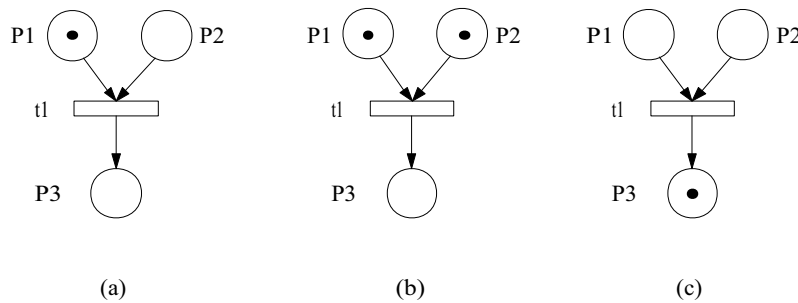


Figura 2.1: Exemplos de sensibilização e disparo de transição em uma rede de Petri.

2.2 Workflow nets

Uma rede de Petri que modela um processo de workflow é uma Workflow net [van der Aalst e van Hee 2004], [van der Aalst 1998]. Uma Workflow net satisfaz as seguintes propriedades [van der Aalst 1998]:

- 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) deve estar em um caminho que se encontra entre o lugar $Start$ e o lugar End .

A Figura 2.2 mostra os elementos de modelagem das WorkFlow nets.

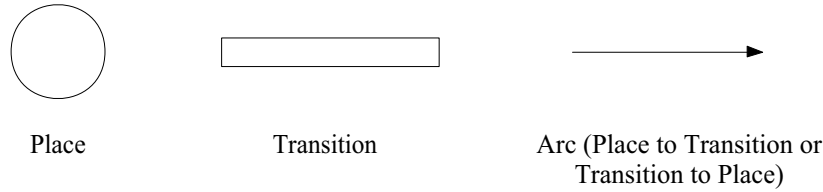


Figura 2.2: Elementos de Modelagem de uma WorkFlow net.

2.2.1 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 [van der Aalst 1998].

Para ilustrar o mapeamento de processos em WorkFlow nets, considera-se o processo de tratamento de reclamações apresentado em [van der Aalst e van Hee 2004]: “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 2.3 mostra a WorkFlow net que representa este processo.

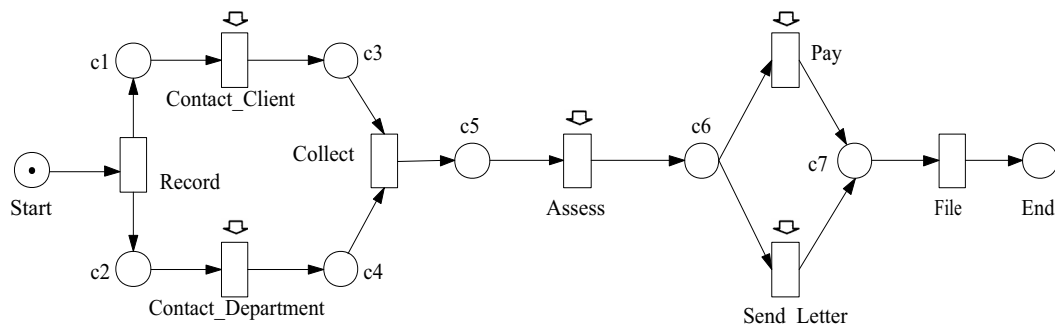


Figura 2.3: WorkFlow net para o processo de tratamento de reclamações e os seus acionamentos.

2.2.2 Roteamentos

De acordo com [van der Aalst e van Hee 2004], 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, ambas as tarefas podem ser executadas sem que o resultado de uma interfira no resultado da outra.
- *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 2.3, as tarefas *Contact_Client* e *Contact_Department* são um exemplo de roteamento paralelo, as tarefas *Collect* e *Assess* são um exemplo de roteamento sequencial e as tarefas *Pay* e *Send_Letter* 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 será substituída por uma tarefa global, como mostrado na Figura 2.4. Na verdade, 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, baseadas no conceito de blocos bem formados¹ [Valette 1979], pode ser utilizada para representar uma rota iterativa através de uma única tarefa. Assim, uma duração máxima será 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.

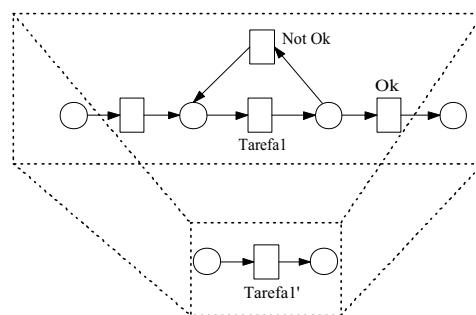


Figura 2.4: Bloco bem formado.

¹do inglês “Well-formed block”

2.2.3 Acionamentos

Um acionamento é uma condição externa que guia a execução de uma tarefa sensibilizada [van der Aalst 1998]. 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 ∇ 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.

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 associada a esta tarefa. Nos demais tipos de tarefa não há alocação de recursos associada.

O processo de tratamento de reclamações mostrado na Figura 2.3 consiste de oito tarefas, das quais três são automaticamente tratadas (*Record*, *Collect*, *File*) e cinco são acionadas por recursos humanos (*Contact_Client*, *Contact_Department*, *Assess*, *Pay* e *Send_Letter*).

2.2.4 Metamodelo das Workflow nets

O metamodelo das Workflow nets é mostrado na Figura 2.5 e será utilizado na transformação proposta no Capítulo 6. Através deste metamodelo, pode-se notar que uma Workflow net é composta de nós (*Nodes*) e arcos (*Arcs*). Nas Workflow nets há dois tipos de nó: as transições (*Transition*) e os lugares (*Place*). Elas possuem também dois tipos de arco:

- *TransitionToPlace*: arcos que ligam transições aos lugares;
- *PlaceToTransition*: arcos que ligam lugares às transições.

O conhecimento destes termos é essencial para o entendimento das transformações apresentadas no Capítulo 6.

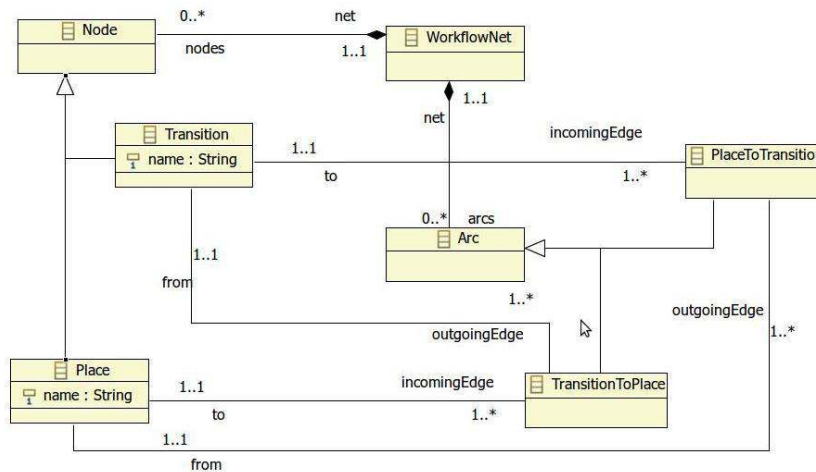


Figura 2.5: Metamodelo das WorkFlow nets.

2.3 Soundness

Soundness 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 workflow e a prova desse critério está relacionada com a análise qualitativa, no contexto das WorkFlow nets.

Em [van der Aalst e van Hee 2004], são apresentados três métodos para provar a propriedade *soundness*: o primeiro método é baseado na construção de grafos de alcançabilidade, o segundo é baseado na análise das propriedades *liveness* e *boundedness* para uma rede “short-circuited” e o terceiro é baseado na substituição de blocos bem formados. Em [van der Aalst 1997], por exemplo, o método para provar *soundness* é baseado na construção de grafos de cobertura, que podem consumir muito tempo em sua construção. O autor também propõe uma nova classe de WorkFlow nets, as “Free-choice WorkFlow nets” que permitem determinar a propriedade *soundness* em tempo polinomial. Em [van der Aalst 1996], são apresentadas as “Well-Structured WorkFlow nets”, que tem um número desejável de propriedades e para as quais a propriedade *soundness* também pode ser verificada em tempo polinomial. Em [Bi e Zhao 2004] é proposto um método baseado em

lógica proposicional para verificar boas propriedades em processos de workflow. Em particular, um formalismo de modelagem de workflow, baseado em atividades, para verificação baseada em lógica é apresentado. Neste caso, a complexidade do algoritmo apresentado para a verificação é $O(n^2)$, mas este algoritmo não é completo e não pode tratar alguns padrões de workflow sobrepostos. Em [Li e Song 2005], um subconjunto das WorkFlow nets clássicas é proposto: as “Normalized WorkFlow nets”. As “Normalized WorkFlow nets” adicionam algumas restrições estruturais aos padrões de roteamento das WorkFlow nets, que garantem a propriedade *soundness*. Entretanto, alguns processos de workflow não podem ser modelados utilizando a definição das “Normalized WorkFlow nets” devido às suas limitações de construção.

2.4 *t*-Time WorkFlow nets

De acordo com [van der Aalst e van Hee 2004], dado um processo modelado por uma rede de Petri, realizar afirmações sobre sua performance é sempre desejado. Para estar apto a responder questões como o tempo esperado de conclusão de um processo e a capacidade de recursos requerida é necessário incluir informações pertinentes sobre o tempo neste modelo de processo. As redes de Petri clássicas não permitem a modelagem do tempo, mas estas podem ser estendidas em relação ao tempo.

As redes de Petri temporais e temporizadas podem associar tempo aos lugares (redes de Petri p-temporais e p-temporizadas) ou às transições (redes de Petri t-temporais e t-temporizadas). Nas redes de Petri p-temporizadas [Sifakis 1977], o tempo é representado por durações (números racionais positivos ou nulo) associadas aos lugares. Nas redes de Petri t-temporizadas [Ramchandani 1974], o tempo é representado por durações (números reais positivos ou nulo) associadas às transições. Nas redes de Petri t-temporais [Merlin 1974], o tempo é representado por um intervalo $[\theta_{min}, \theta_{max}]$ associado a cada transição, de forma que este intervalo de tempo corresponde a uma duração de sensibilização. Assim, o intervalo $[8, 12]$ indica que a transição irá disparar pelo menos oito unidades de tempo após a transição ter sido sensibilizada e no máximo doze unidades de tempo após a sensibilização desta transição. Nas redes de Petri p-temporais [Khansa 1997], é associado um intervalo de tempo $[\theta_{min}, \theta_{max}]$ aos lugares.

No contexto das redes de Petri t-temporais, uma *t*-Time WorkFlow net será então uma WorkFlow net estendida com intervalos de tempo associados às transições, já que a execução das tarefas ficará associada às transições do modelo.

Definição 2.2. (t-Time WorkFlow net) Uma *t*-Time WorkFlow net N é uma quádrupla (P, T, F, I) tal que:

- (P, T, F) é uma WorkFlow net, onde:
 - P é um conjunto finito de lugares,

- T é um conjunto finito de transições e
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto de arcos (relação de fluxo).
- I é uma aplicação que associa a cada transição $t \in T$ um intervalo de sensibilização $I(t) = [\theta_{\min(t)}, \theta_{\max(t)}]$, onde $\theta_{\min(t)}$ representa o tempo mínimo de disparo da transição e $\theta_{\max(t)}$ representa o tempo máximo de disparo da transição t .

2.5 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, as WorkFlow nets, a propriedade *soundness* e as t-Time WorkFlow nets.

Capítulo 3

Redes de Petri e Lógica Linear

Este capítulo tem por objetivo apresentar a lógica linear e sua relação com as redes de Petri. Para tanto, a Seção 3.1 apresenta brevemente a lógica linear e suas diferenças principais em relação à lógica clássica. Já a Seção 3.2 mostra como representar uma rede de Petri através de fórmulas da lógica linear. Na Seção 3.3 é apresentado o sistema de dedução linear, assim como as regras do cálculo de sequentes. Finalmente, as árvores de prova canônica com cálculo de datas são apresentadas na Seção 3.4.

3.1 Lógica Linear

A lógica linear foi proposta em 1987 por Girard em [Girard 1987]. Na lógica linear, as proposições são consideradas como recursos que são consumidos ou produzidos a cada mudança de estado [Pradin-Chezalviel et al. 1999], enquanto que, na lógica clássica as proposições podem tomar valores *verdadeiro* ou *falso*. As principais diferenças entre a lógica clássica e a lógica linear são a inexistência das regras de enfraquecimento e de contração.

De acordo com a *Regra de Enfraquecimento*, se A implica B , então A e C implicam B [Gochet e Gribomont 1990]. Considerando um sistema onde A , B e C são recursos, a regra de enfraquecimento mostraria que recursos, neste caso C , poderiam aparecer e desaparecer sem nenhuma influência. Assim, se a partir de um recurso A pode-se produzir um recurso B (consumindo o recurso A), então o aparecimento de um recurso C altera o sistema, o que não é mostrado pela regra de enfraquecimento da lógica clássica [Soares 2004].

De acordo com a *Regra de Contração*, se A e A implicam B , então A implica B [Gochet e Gribomont 1990]. No contexto da lógica linear, utilizar esta regra seria o mesmo que aceitar que, havendo a necessidade de dois recursos A para produzir um recurso B , um desses recursos poderia ser dispensado. O que não é verdadeiro neste contexto, pois, se houver apenas um recurso A , B não poderá mais ser produzido.

Outra importante diferença entre a lógica clássica e a lógica linear é que, enquanto

na lógica clássica uma verdade é dita estável, na lógica linear o mesmo não acontece. Por exemplo, supondo que A e B sejam duas proposições da lógica clássica e que A seja verdadeira. Então, se A implica B , então B é deduzido. Assim, a proposição A continua sendo verdadeira mesmo após ter sido usada na dedução de B . Na lógica linear isso não é possível, pois se A foi consumido para produzir B , então A não é mais válido [Soares 2004].

3.1.1 Conectivos da Lógica Linear

A lógica linear introduz novos conectivos, como os conectivos “*par*”($\wp$), “*times*”($\otimes$), “*with*”($\&$), “*plus*”($\oplus$), *implicação linear* ($\multimap$), “*of course*” (!) e “*why not*” (?) [Girard 1987], [Pradin-Chezalviel et al. 1999].

Estes conectivos estão divididos em três grupos:

- Os conectivos multiplicativos: *implicação linear* ($\multimap$), “*times*”($\otimes$) e “*par*”($\wp$).
- Os conectivos aditivos: “*with*”($\&$) e “*plus*”($\oplus$).
- Os conectivos exponenciais: “*of course*” (!) e “*why not*” (?).

No contexto deste trabalho, apenas dois conectivos da lógica linear serão explorados:

- O conectivo *times*, denotado pelo símbolo $\otimes$, representa a disponibilidade simultânea de recursos. Por exemplo, $A \otimes B$ representa a disponibilidade simultânea dos recursos A e B .
- O conectivo *implicação linear*, denotado pelo símbolo $\multimap$, representa uma mudança de estado. Por exemplo, $A \multimap B$ denota que consumindo A , B é produzido. Deve-se notar que após a produção de B , A não estará mais disponível.

Na Seção 3.3 são apresentadas as regras de dedução usando estes dois conectivos.

3.2 Representação de Redes de Petri usando Lógica Linear

Apesar das redes de Petri poderem ser reescritas utilizando lógica linear, as duas teorias não são equivalentes, sendo que cada uma tem suas vantagens específicas [Roman Champagnat 2000]. As redes de Petri permitem o cálculo de invariantes de lugar e de transição, enquanto a lógica linear trata claramente cenários que envolvem a produção e consumo de recursos [Soares 2004].

A tradução de uma rede de Petri em fórmulas da lógica linear é apresentada em [Pradin-Chezalviel et al. 1999].

Uma marcação M é um monômio em $\otimes$, ou seja, uma marcação é representada por $M = A_1 \otimes A_2 \otimes \dots \otimes A_k$ onde A_i são nomes de lugares. Por exemplo, considerando a rede de Petri da Figura 3.1(a), tem-se que sua marcação inicial $M = P1$. Já a marcação inicial da rede de Petri da Figura 3.1(b) é dada por $M = P1 \otimes P2$.

Uma transição é uma expressão da forma $M_1 \multimap M_2$ onde M_1 e M_2 são marcações. Por exemplo, considerando a transição $t1$ da rede de Petri mostrada na Figura 3.1(a), tem-se que $t1 = P1 \multimap P2$. Para a transição $t1$ da rede de Petri da Figura 3.1(b), tem-se que $t1 = P1 \otimes P2 \multimap P3$. Já para a transição $t1$ da rede de Petri da Figura 3.1(c), tem-se que $t1 = P1 \multimap P2 \otimes P3$.

Um sequente $M, t_i \vdash M'$ representa um cenário onde M e M' são, respectivamente, a marcação inicial e final e t_i é uma lista de transições não ordenadas [Julia e Soares 2003]. Por exemplo, considerando a rede de Petri da Figura 3.1(a) e que sua marcação final seja $P2$, tem-se o seguinte sequente linear $P1, P1 \multimap P2 \vdash P2$. Para a rede de Petri da Figura 3.1(b), considerando que sua marcação final seja $P3$, tem-se o sequente $P1 \otimes P2, P1 \otimes P2 \multimap P3 \vdash P3$. Considerando a rede de Petri da Figura 3.1(c) e que sua marcação final seja $P2 \otimes P3$, tem-se o sequente $P1, P1 \multimap P2 \otimes P3 \vdash P2 \otimes P3$.

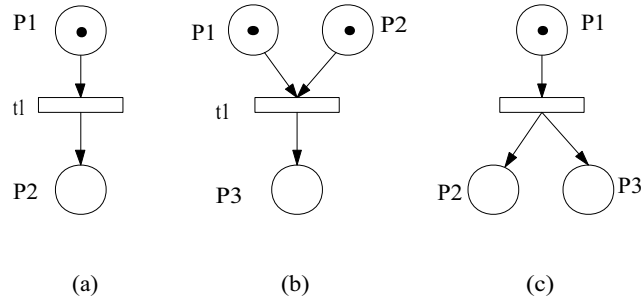


Figura 3.1: Redes de Petri para a exemplificação da tradução de redes de Petri em fórmulas da lógica linear.

3.3 Cálculo de Sequentes

O sistema de dedução linear é similar ao sistema de dedução de Gentzen, proposto em 1934 [Gochet e Gribomont 1990]. Um sequente linear tem a forma $\Gamma \vdash \Delta$, onde Γ e Δ são conjuntos finitos de fórmulas, isto é, $\Gamma = \Gamma_1, \Gamma_2, \dots, \Gamma_n$ e $\Delta = \Delta_1, \Delta_2, \dots, \Delta_n$. O símbolo Γ é o antecedente da fórmula e o símbolo Δ o conseqüente.

Um sequente pode ser provado através de aplicações sucessivas de regras do cálculo de sequentes. De acordo com [Girault 1997], há equivalência entre a prova de sequentes da lógica linear e o problema de alcançabilidade em uma rede de Petri.

Uma árvore de prova da lógica linear é construída para provar se um dado sequente linear é ou não sintaticamente válido. A árvore de prova é lida de baixo para cima (bottom-up) e termina quando todas as folhas forem sequentes identidade (sequentes do tipo $A \vdash A$) [Julia e Soares 2003], considerando o caso em que o sequente é válido.

O presente trabalho considera apenas algumas regras da lógica linear. Assim, somente estas regras serão explicadas e terão suas aplicações exemplificadas. As demais regras do cálculo de seqüentes da lógica linear podem ser encontradas em [Girard 1987] e em [Girard 1995]. As regras aqui apresentadas são utilizadas na construção das árvores de prova canônica no contexto das redes de Petri e, conseqüentemente, no contexto das WorkFlow nets. Para tanto, considere que F , G e H são fórmulas e que Γ e Δ são blocos de fórmulas da lógica linear. A seguir seguem as regras [Riviere et al. 2001]:

- A regra $\multimap_L$, dada por $\frac{\Gamma \vdash F \quad \Delta, G \vdash H}{\Gamma, \Delta, F \multimap G \vdash H} \multimap_L$, expressa o disparo de uma transição e gera dois seqüentes, tal que o seqüente à direita representa o subsequente restante a ser provado e o seqüente à esquerda representa as fichas consumidas pelo disparo da transição.

Por exemplo, considerando o disparo da transição $t_1 = P1 \multimap P2 \otimes P3$ da rede de Petri mostrada na Figura 3.2, dois seqüentes são gerados: $P1 \vdash P1$, representando as fichas consumidas por esse disparo, e o subsequente remanescente, do qual a marcação $P2 \otimes P3$ fará parte, ou seja: $\frac{P1 \vdash P1 \quad P2 \otimes P3, P2 \multimap P4, t_3, t_4 \vdash P6}{P1, P1 \multimap P2 \otimes P3, t_2, t_3, t_4 \vdash P6} \multimap_L$.

- A regra $\otimes_L$, dada por $\frac{\Gamma, F, G \vdash H}{\Gamma, F \otimes G \vdash H} \otimes_L$, transforma uma marcação em uma lista de átomos.

Por exemplo, a marcação $P2 \otimes P3$ gerada pelo disparo da transição $t_1 = P1 \multimap P2 \otimes P3$ da rede de Petri mostrada na Figura 3.2 utilizará a regra $\otimes_L$ para ser transformada em uma lista de átomos $P2, P3$, ou seja: $\frac{P2, P3, t_2, t_3, t_4 \vdash P6}{P2 \otimes P3, t_2, t_3, t_4 \vdash P6} \otimes_L$.

- A regra $\otimes_R$, dada por $\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Delta, \Gamma \vdash F \otimes G} \otimes_R$, transforma um seqüente do tipo $A, B \vdash A \otimes B$ em dois seqüentes identidade $A \vdash A$ e $B \vdash B$.

Por exemplo, considerando o disparo da transição $t_4 = P4 \otimes P5 \multimap P6$ da rede de Petri da Figura 3.2, o seqüente que representa as fichas consumidas pelo disparo desta transição, $P4, P5 \vdash P4 \otimes P5$, deverá ser provado utilizando a regra $\otimes_R$, ou seja: $\frac{P4 \vdash P4 \quad P5 \vdash P5}{P4, P5 \vdash P4 \otimes P5} \otimes_R$.

Para a exemplificação da construção de uma árvore de prova canônica da lógica linear, considere a rede de Petri da Figura 3.2. Transformando as transições desta rede de Petri em fórmulas da lógica linear, tem-se que:

$$\begin{aligned} t_1 &= P1 \multimap P2 \otimes P3 \\ t_2 &= P2 \multimap P4 \\ t_3 &= P3 \multimap P5 \text{ e} \\ t_4 &= P4 \otimes P5 \multimap P6. \end{aligned}$$

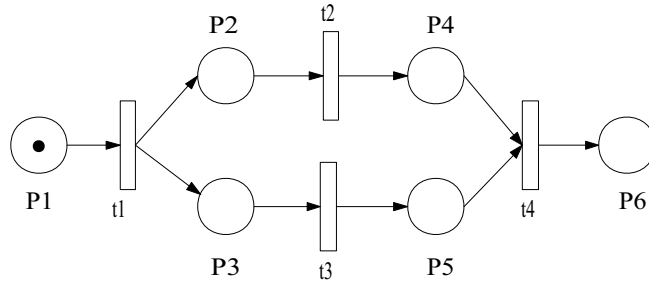


Figura 3.2: Rede de Petri para exemplificação da construção de uma árvore de prova canônica da lógica linear.

A marcação inicial desta rede de Petri é apenas $P1$. Assim, o sequeute linear a ser provado é dado por $P1, t_1, t_2, t_3, t_4 \vdash P6$, onde $P1$ e $P6$ são, respectivamente, a marcação inicial e final da rede de Petri da Figura 3.2. Aplicando as regras do cálculo de sequeutes a este sequeute linear, é possível provar se o mesmo é ou não um sequeute sintaticamente válido.

A árvore de prova se segue:

$$\begin{array}{c}
 \frac{\frac{P4 \vdash P4 \quad P5 \vdash P5}{P4, P5 \vdash P4 \otimes P5} \otimes_R \quad P6 \vdash P6}{\quad} \multimap_L \\
 \frac{P3 \vdash P3 \quad P4, P5, P4 \otimes P5 \multimap P6 \vdash P6}{\quad} \multimap_L \\
 \frac{P2 \vdash P2 \quad P3, P4, P3 \multimap P5, t_4 \vdash P6}{\quad} \multimap_L \\
 \frac{\quad}{P2, P3, P2 \multimap P4, t_3, t_4 \vdash P6} \otimes_L \\
 \frac{P1 \vdash P1 \quad P2 \otimes P3, P2 \multimap P4, t_3, t_4 \vdash P6}{\quad} \multimap_L \\
 P1, P1 \multimap P2 \otimes P3, t_2, t_3, t_4 \vdash P6
 \end{array}$$

Como todos os nós folha da árvore de prova acima são sequeutes identidade, ou seja, sequeutes do tipo $A \vdash A$, tem-se que o sequeute linear $P1, t_1, t_2, t_3, t_4 \vdash P6$ é sintaticamente válido.

3.4 Árvore de Prova Canônica com Cálculo de Datas

No contexto das redes de Petri t-temporais, em uma árvore de prova da lógica linear, cada disparo de transição pode gerar uma data simbólica associada a cada átomo (token), como mostrado em [Riviere et al. 2001].

No presente trabalho, D_i denota uma data e d_i uma duração associada a um disparo de uma transição (t_i). Um par (D_p, D_c) será associado a cada átomo da árvore de prova, tal que D_p e D_c representam, respectivamente, as datas de produção e consumo de um átomo. O cálculo de datas em árvores de prova canônica é dado pelos seguintes passos:

- Determinar uma data de produção D_i para todas as fichas da marcação inicial;

- para cada instância da regra $\multimap_L$, calcule a data de disparo desta transição: isto é igual ao maior valor de data de produção dos átomos consumidos por esta transição, acrescido pela duração de sensibilização d_i associada à transição considerada;
- atualizar as datas de todos os átomos que foram consumidos e produzidos.

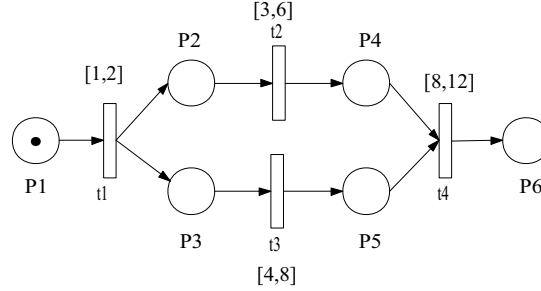


Figura 3.3: Rede de Petri t-temporal para exemplificação da construção de uma árvore de prova canônica da lógica linear com cálculo de datas.

Por exemplo, considerando a rede de Petri t-temporal da Figura 3.3, as fórmulas da lógica linear para cada uma de suas transições (definidas na Seção 3.3), sua marcação inicial $P1$ e que $Seq = D_1 + d_1 + \max(d_2, d_3) + d_4$, tem-se a seguinte árvore de prova com cálculo de datas:

$$\begin{array}{c}
 \frac{\frac{P4(D_1+d_1+d_2, Seq) \vdash P4 \quad P5(D_1+d_1+d_3, Seq) \vdash P5}{P4(D_1+d_1+d_2, Seq), P5(D_1+d_1+d_3, Seq) \vdash P4 \otimes P5} \otimes_R \quad P6(Seq, \cdot) \vdash P6}{P3(D_1+d_1, D_1+d_1+d_3) \vdash P3 \quad P4(D_1+d_1+d_2, \cdot), P5(D_1+d_1+d_3, \cdot), P4 \otimes P5 \multimap P6 \vdash P6} \multimap_L \\
 \frac{P2(D_1+d_1, D_1+d_1+d_2) \vdash P2 \quad P3(D_1+d_1, \cdot), P4(D_1+d_1+d_2, \cdot), P3 \multimap P5, t_4 \vdash P6}{P2(D_1+d_1, \cdot), P3(D_1+d_1, \cdot), P2 \multimap P4, t_3, t_4 \vdash P6} \multimap_L \\
 \frac{P1(D_1, D_1+d_1) \vdash P1 \quad P2(D_1+d_1, \cdot) \otimes P3(D_1+d_1, \cdot), P2 \multimap P4, t_3, t_4 \vdash P6}{P1(D_1, \cdot), P1 \multimap P2 \otimes P3, t_2, t_3, t_4 \vdash P6} \multimap_L
 \end{array}$$

A Tabela 3.1 mostra as datas simbólicas de produção e consumo de cada átomo da rede de Petri t-temporal da Figura 3.3.

Em um modelo de rede de Petri t-temporal, toda duração de sensibilização d_i de uma transição t_i tem um valor que pertence a um intervalo de tempo $\Delta_i = [\delta_{i_{min}}, \delta_{i_{max}}]$. Logo, uma vez que as datas simbólicas computadas dependem de d_i , seus domínios também serão em função de intervalos de tempo. A Tabela 3.2 mostra os intervalos de datas simbólicas de produção e consumo de cada átomo da rede de Petri t-temporal da Figura 3.3.

Átomo	Data de Produção	Data de Consumo
$P1$	D_1	D_1+d_1
$P2$	D_1+d_1	$D_1+d_1+d_2$
$P3$	D_1+d_1	$D_1+d_1+d_3$
$P4$	$D_1+d_1+d_2$	$D_1+d_1+max(d_2,d_3)+d_4$
$P5$	$D_1+d_1+d_3$	$D_1+d_1+max(d_2,d_3)+d_4$
$P6$	$D_1+d_1+max(d_2,d_3)+d_4$	<i>desconhecido</i>

Tabela 3.1: Datas simbólicas de produção e consumo dos átomos da rede de Petri t-temporal da Figura 3.3.

Átomo	Data de Produção	Data de Consumo
$P1$	$[D_{1min}, D_{1max}]$	$[D_{1min}+d_{1min}, D_{1max}+d_{1max}]$
$P2$	$[D_{1min}+d_{1min}, D_{1max}+d_{1max}]$	$[D_{1min}+d_{1min}+d_{2min}, D_{1max}+d_{1max}+d_{2max}]$
$P3$	$[D_{1min}+d_{1min}, D_{1max}+d_{1max}]$	$[D_{1min}+d_{1min}+d_{3min}, D_{1max}+d_{1max}+d_{3max}]$
$P4$	$[D_{1min}+d_{1min}+d_{2min}, D_{1max}+d_{1max}+d_{2max}]$	$[D_{1min}+d_{1min}+max(d_{2min}, d_{3min})+d_{4min}, D_{1max}+d_{1max}+max(d_{2max}, d_{3max})+d_{4max}]$
$P5$	$[D_{1min}+d_{1min}+d_{3min}, D_{1max}+d_{1max}+d_{3max}]$	$[D_{1min}+d_{1min}+max(d_{2min}, d_{3min})+d_{4min}, D_{1max}+d_{1max}+max(d_{2max}, d_{3max})+d_{4max}]$
$P6$	$[D_{1min}+d_{1min}+max(d_{2min}, d_{3min})+d_{4min}, D_{1max}+d_{1max}+max(d_{2max}, d_{3max})+d_{4max}]$	<i>desconhecido</i>

Tabela 3.2: Intervalos de datas simbólicas de produção e consumo dos átomos da rede de Petri t-temporal da Figura 3.3.

3.5 Considerações Finais do Capítulo

Este capítulo apresentou a lógica linear e sua relação com as redes de Petri, mostrando brevemente a lógica linear e suas diferenças principais em relação à lógica clássica, como representar uma rede de Petri através de fórmulas da lógica linear, o sistema de dedução linear e as árvores de prova canônica com cálculo de datas.

Capítulo 4

ATL e Diagramas de Atividades da UML

O presente capítulo tem por objetivo apresentar a linguagem de transformação ATL (*Atlas Transformation Language*) e os Diagramas de Atividades da UML. Assim, a Seção 4.1 apresenta a referida linguagem. Os Diagramas de Atividades da UML e seu metamodelo são definidos na Seção 4.2.

4.1 ATL - *Atlas Transformation Language*

No contexto da Engenharia Dirigida por Modelos, os modelos são os principais artefatos de desenvolvimento e as transformações de modelos são as operações mais importantes que são aplicadas aos modelos [Jouault et al. 2008]. Atualmente, há muitas linguagens de transformação disponíveis, como a linguagem de transformação ATLAS (ATL), desenvolvida pelo Grupo ATLAS (INRIA & LINA). No contexto da Engenharia Dirigida por Modelos, a ATL provê maneiras de produzir um conjunto de modelos objetivo a partir de um conjunto de modelos fonte [Atlas_Group].

A ATL é um modelo híbrido de transformação que contém uma mistura de construções declarativas e imperativas. As transformações ATL são unidirecionais, isto é, os modelos fonte são apenas para leitura e os modelos objetivo são apenas de escrita.

Durante a execução de uma transformação, o modelo fonte deve ser percorrido mas alterações não são permitidas. O modelo objetivo não pode ser percorrido e é gerado pela transformação. Um exemplo de transformação ATL, que, neste caso, transforma um elemento *Action* de um Diagrama de Atividades da UML em um elemento *Transition* de uma WorkFlow net é apresentado na Figura 4.1.

Um módulo ATL contém uma seção de cabeçalhos obrigatória e as regras de transformação. A seção de cabeçalhos (linhas 1 e 2 da transformação ATL mostrada na Figura 4.1) dá o nome do módulo de transformação e declara os modelos fonte e objetivo. Os modelos

```
1 module ActivityDiagram2WorkflowNet;  
2 create OUT : WorkflowNet from IN : ActivityDiagram;  
3 rule Action2Transition {  
4     from action : ActivityDiagram!Action  
5  
6     to transition : WorkflowNet!Transition (  
7         name <- action.name,  
8         net <- action.ad  
9     )  
10 }
```

Figura 4.1: Exemplo de transformação em ATL.

fonte e objetivo são delimitados por seus metamodelos. As regras declarativas da ATL especificam relações entre elementos do modelo fonte e do modelo objetivo. O nome da regra de transformação é dado antes da própria regra. O padrão fonte (“from”) de uma regra (linha 4 da transformação ATL mostrada na Figura 4.1) especifica um conjunto de elementos fonte e uma guarda opcional é dada através de uma expressão booleana em OCL (*Object Constraint Language*). Um padrão fonte é avaliado por uma série de “casamentos” no modelo fonte. O padrão objetivo (“to”) (linhas de 6–8 da transformação ATL mostrada na Figura 4.1) é composto de um conjunto de elementos do modelo objetivo. Cada um desses elementos especifica um elemento objetivo do metamodelo objetivo e um conjunto de ligações. Uma ligação se refere a uma característica do elemento (por exemplo, um atributo, uma referência ou uma associação) e especifica uma expressão cujo valor é usado para iniciar a característica.

Em alguns casos, pode ser necessária a utilização de algoritmos de transformação complexos e pode ser difícil especificá-los de maneira declarativa. Para isso, a ATL provê construções imperativas. No caso desse trabalho, apenas construções declarativas foram utilizadas.

4.2 Diagramas de Atividades da UML

A UML (*Unified Modeling Language*) é uma linguagem de modelagem amplamente utilizada no contexto da Engenharia de Software. Os diagramas da UML estão divididos em três categorias [OMG 2008]:

- Diagramas Estruturais: diagrama de objetos, diagrama de classes diagrama de componentes, diagrama de implantação, diagrama de pacotes e diagrama de estruturas compostas.
- Diagramas Comportamentais: diagrama de caso de uso, diagrama de máquinas de estado e diagrama de atividades.

- Diagramas de Interação: diagrama de sequência, diagrama de visão geral de interação, diagrama de comunicação (ou colaboração) e diagrama de tempo.

Os Diagramas de Atividades da UML na versão 1.3 eram tratados como casos especiais das máquinas de estado [Tričković 2000], tal que todos os estados eram estados ação e as transições eram disparadas pela finalização de seus estados fonte. Já a partir da versão 2.0 da UML, estes diagramas foram redesenhados utilizando uma semântica parecida com a das redes de Petri ao invés de máquinas de estado [OMG 2008], [Störrle 2005]. Além de ter outras aplicações, estes diagramas são utilizados na modelagem de processos de workflow.

Os elementos de modelagem dos Diagramas de Atividades da UML são definidos no documento “OMG UML Superstructure” [OMG 2008]. A Figura 4.2 mostra os elementos de modelagem de um Diagrama de Atividades da UML que serão abordados neste trabalho. Os demais elementos não são considerados nesta abordagem. Além disso, nesta abordagem, algumas restrições são impostas aos Diagramas de Atividades da UML. Por exemplo, apenas um *Initial Node* e um *Final Node* podem aparecer no Diagrama de Atividades da UML. Todas as atividades (Action) precisam ter pelo menos uma aresta (Control Flow) saindo delas de forma que esta aresta tenha como alvo uma atividade diferente daquela que é a atividade predecessora da atividade da qual ela sai.

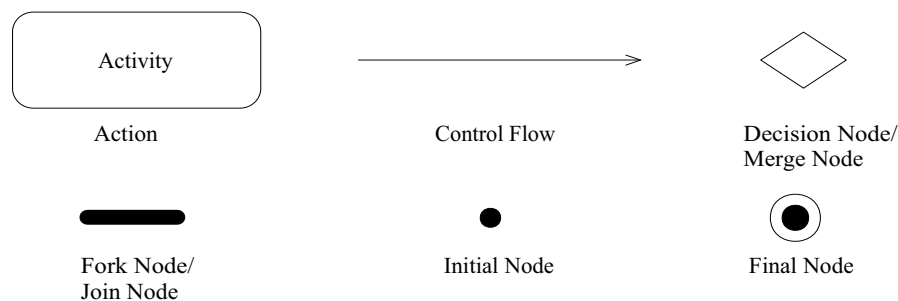


Figura 4.2: Elementos de Modelagem de um Diagrama de Atividades da UML.

4.2.1 Metamodelo dos Diagramas de Atividades

Os metamodelos dos Diagramas de Atividades da UML e das WorkFlow nets serão utilizados na transformação entre estes modelos, apresentada no Capítulo 6 deste trabalho. A Figura 4.3 mostra graficamente o metamodelo dos Diagramas de Atividades. O metamodelo das WorkFlow nets foi apresentado na Figura 2.5.

A maior diferença entre o Diagrama de Atividades da UML proposto pela OMG (*Object Management Group*) [OMG 2008] e o proposto neste trabalho é que a abordagem proposta não considera nem os *Object Nodes* nem os *Object Flows*, que aparecem no metamodelo proposto pela OMG.

A título de exemplificação, a Figura 4.4(a) apresenta um pequeno Diagrama de Atividades da UML e a Figura 4.4(b) apresenta seu respectivo Diagrama de Objetos, de acordo com o metamodelo dos Diagramas de Atividades da UML mostrado na Figura 4.3

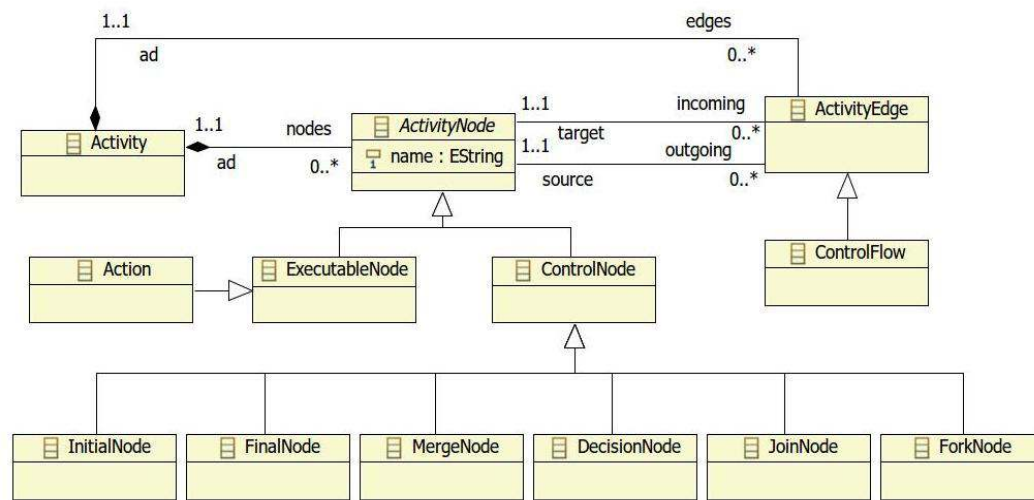


Figura 4.3: Metamodelo dos Diagramas de Atividades da UML.

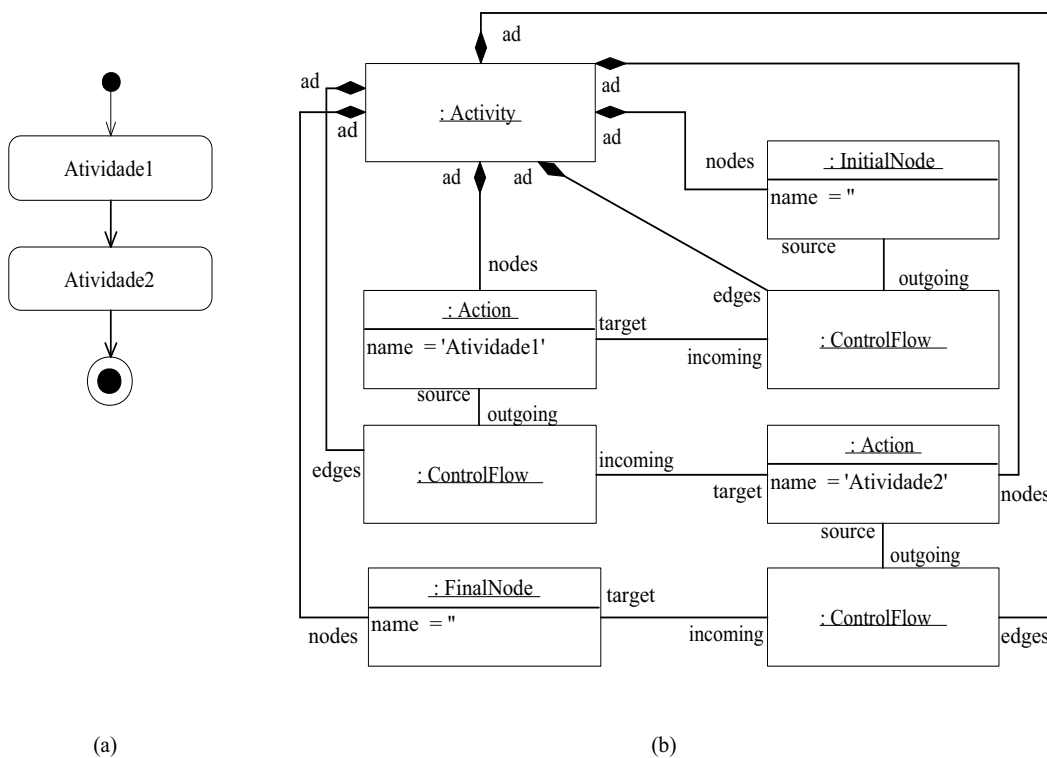


Figura 4.4: Exemplo de Diagrama de Atividades da UML com seu respectivo Diagrama de Objetos. (a) Diagrama de Atividades. (b) Diagrama de Objetos.

4.3 Considerações Finais do Capítulo

O presente capítulo apresentou a linguagem de transformação ATL (*Atlas Transformation Language*) e os Diagramas de Atividades da UML.

Capítulo 5

Análise Qualitativa e Quantitativa de WorkFlow nets

Este capítulo tem por objetivo apresentar uma abordagem para a análise qualitativa e quantitativa de WorkFlow nets. A análise qualitativa refere-se à verificação da propriedade *soundness* e é apresentada na Seção 5.1. Já a análise quantitativa de WorkFlow nets preocupa-se com o planejamento de recursos para cada atividade de um processo de workflow mapeado em uma t-Time WorkFlow net, sendo apresentada na Seção 5.2.

5.1 Análise Qualitativa: Verificação da Propriedade *Soundness*

No contexto das WorkFlow nets, o objetivo principal da análise qualitativa é provar o critério de corretude definido para as WorkFlow nets que é denominado *soundness*.

Para isso, é necessário provar sequentes da lógica linear que representem a WorkFlow net e analisar as árvores de prova construídas para provar estes sequentes.

Inicialmente, se a WorkFlow net possuir mais de uma rota (ou seja, um ou mais roteamentos condicionais), é necessário construir e provar um sequente linear para cada cenário da WorkFlow net. Um cenário corresponde a uma rota bem definida mapeada na WorkFlow net. Por exemplo, na Figura 2.3, há dois cenários diferentes: o primeiro cenário, Sc_1 , onde a tarefa *Pay* será executada, isto é, a transição *Pay* será disparada, e o segundo cenário, Sc_2 , onde a tarefa *Send_Letter* será executada.

Assim, para a WorkFlow net da Figura 2.3, dois sequentes da lógica linear deverão ser provados, sendo que cada sequente linear considera apenas uma rota da WorkFlow net analisada. Por exemplo, para o cenário Sc_1 , é necessário provar o sequente:

$$Start, Record, Contact_Client, Contact_Department, Collect, Assess, Pay, File \vdash End$$

e para o cenário Sc_2 é necessário provar o sequente:

$Start, Record, Contact_Client, Contact_Department, Collect, Assess, Send_Letter, File \vdash End$

onde:

$$Record = t_1 = Start \multimap c1 \otimes c2,$$

$$Contact_Client = t_2 = c1 \multimap c3,$$

$$Contact_Department = t_3 = c2 \multimap c4,$$

$$Collect = t_4 = c3 \otimes c4 \multimap c5,$$

$$Assess = t_5 = c5 \multimap c6,$$

$$Pay = t_6 = c6 \multimap c7,$$

$$Send_Letter = t_7 = c6 \multimap c7 \text{ e}$$

$$File = t_8 = c7 \multimap End.$$

Deve-se notar que em cada sequente linear tem apenas um átomo *Start*, que representa a marcação inicial da Workflow net. Isso acontece uma vez que, nesta abordagem, para analisar o critério de corretude *soundness*, apenas uma ficha no lugar *Start* é necessária e suficiente. Vale destacar que apenas uma ficha no lugar *Start* é necessária e suficiente pois, se não houver uma ficha no lugar *Start*, não haverá no sequente linear um átomo *Start* e a prova sintática do sequente não será realizada e se houver mais de uma ficha neste lugar, as demais fichas não serão utilizadas e ficarão remanescentes na árvore de prova. Assim, para a construção dos sequentes da lógica linear, sempre será considerada apenas uma ficha no lugar *Start*.

A árvore de prova da lógica linear utilizada no contexto desse trabalho pode terminar de três formas:

- Quando todos as folhas da árvore de prova forem sequentes identidade;
- Quando o átomo *End* for produzido (ou seja, quando o sequente $End \vdash End$ aparecer na árvore de prova);
- Ou quando não houver nenhuma regra do cálculo de sequentes que possa ser aplicada.

Assim, a árvore de prova utilizada nesta abordagem é decidível, uma vez que cada transição da Workflow net aparece no máximo uma vez no sequente linear. Consequentemente, se o número de transições de uma Workflow net é finito, a construção da árvore de prova é decidível.

Alguns provadores de teoremas da lógica linear automáticos, como LinTAP [Mantel e Otten 1999], Linprove [Tammet 1994] e llprover [Tamura 1997], podem ser adaptados e utilizados para automatizar a prova dos sequentes da lógica linear propostos neste trabalho.

A Proposição 5.1 mostra que a complexidade para provar um sequente linear utilizado no contexto das Workflow nets é linear.

Proposição 5.1. *A complexidade de tempo para provar um sequente linear que representa um cenário de uma WorkFlow net é $O(n)$, onde n é o número de transições da WorkFlow net.*

Prova. Se um sequente linear $M, t_i \vdash M'$ representa um cenário onde M e M' são, respectivamente, a marcação inicial e a marcação final, e se t_i é uma lista de i transições não ordenadas, tal que $i \leq n$, onde n é o número de transições da WorkFlow net analisada. Então, no pior caso, para um dado cenário de uma WorkFlow net, todas as transições t_i da WorkFlow net aparecerão como expressões da lógica linear da forma $c1 \otimes c2 \multimap c3 \otimes c4$. Então, as três regras $\multimap_L$, $\otimes_R$ e $\otimes_L$ serão aplicadas para provar cada disparo de transição. A regra $\multimap_L$ irá representar o disparo da transição, a regra $\otimes_R$ será utilizada para provar o sequente à esquerda, gerado pelo disparo da transição e a regra $\otimes_L$ será utilizada para transformar parte do subsequente remanescente a ser provado em uma lista de átomos. Assim, a complexidade de tempo para provar um sequente linear que representa um cenário será $O(3n) = O(n)$, onde n é o número de transições da WorkFlow net analisada. ■

Para provar os sequentes dos cenários Sc_1 e Sc_2 acima definidos, árvores de prova da lógica linear, sem cálculo de datas serão utilizadas. Na sequência são mostradas as árvores de prova para os cenários Sc_1 e Sc_2 .

Árvore de prova para o cenário Sc_1 :

$$\begin{array}{c}
 \frac{c7 \vdash c7 \quad \text{End} \vdash \text{End}}{\multimap_L} \\
 \frac{c6 \vdash c6 \quad c7, c7 \multimap \text{End} \vdash \text{End}}{\multimap_L} \\
 \frac{c5 \vdash c5 \quad c6, c6 \multimap c7, t_8 \vdash \text{End}}{\multimap_L} \\
 \frac{\frac{c3 \vdash c3 \quad c4 \vdash c4}{c3, c4 \vdash c3 \otimes c4} \otimes_R \quad c5, c5 \multimap c6, t_6, t_8 \vdash \text{End}}{\multimap_L} \\
 \frac{c2 \vdash c2 \quad c3, c4, c3 \otimes c4 \multimap c5, t_5, t_6, t_8 \vdash \text{End}}{\multimap_L} \\
 \frac{c1 \vdash c1 \quad c2, c3, c2 \multimap c4, t_4, t_5, t_6, t_8 \vdash \text{End}}{\multimap_L} \\
 \frac{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash \text{End}}{\otimes_L} \\
 \frac{\text{Start} \vdash \text{Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash \text{End}}{\multimap_L} \\
 \text{Start}, \text{Start} \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_6, t_8 \vdash \text{End}
 \end{array}$$

Árvore de prova para o cenário Sc_2 :

$$\begin{array}{c}
\frac{c7 \vdash c7 \quad \text{End} \vdash \text{End}}{\rightarrow_L} \\
\frac{c6 \vdash c6 \quad c7, c7 \multimap \text{End} \vdash \text{End}}{\rightarrow_L} \\
\frac{c5 \vdash c5 \quad c6, c6 \multimap c7, t_8 \vdash \text{End}}{\rightarrow_L} \\
\frac{\frac{c3 \vdash c3 \quad c4 \vdash c4}{c3, c4 \vdash c3 \otimes c4} \otimes_R \quad c5, c5 \multimap c6, t_7, t_8 \vdash \text{End}}{\rightarrow_L} \\
\frac{c2 \vdash c2 \quad c3, c4, c3 \otimes c4 \multimap c5, t_5, t_7, t_8 \vdash \text{End}}{\rightarrow_L} \\
\frac{c1 \vdash c1 \quad c2, c3, c2 \multimap c4, t_4, t_5, t_7, t_8 \vdash \text{End}}{\rightarrow_L} \\
\frac{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash \text{End}}{\otimes_L} \\
\frac{\text{Start} \vdash \text{Start} \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash \text{End}}{\rightarrow_L} \\
\text{Start}, \text{Start} \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_7, t_8 \vdash \text{End}
\end{array}$$

Quando as árvores de prova para todos os cenários da Workflow net analisada estão construídas, elas devem ser analisadas seguindo os seguintes passos:

1. Para cada árvore de prova construída verificar se:
 - (a) Apenas um átomo *End* foi produzido (isto é representado, na árvore de prova, pelo sequente identidade $\text{End} \vdash \text{End}$). Este fato prova o primeiro requisito para a verificação da propriedade *soundness*, isto é, que apenas uma ficha aparece no lugar *End* para o caso tratado.
 - (b) Quando a prova está finalizada, não há nenhum átomo disponível para consumo na árvore de prova. Este fato prova que todos os lugares da Workflow net estão vazios para este caso quando sua ficha chega no lugar *End*, ou seja, o segundo requisito para afirmar que uma Workflow net é *sound*.
2. Considerando todos os cenários $Sc_1, Sc_2, \dots, Sc_n$ da Workflow net analisada, cada transição $t \in T$ precisa aparecer em um cenário, pelo menos. Isto prova que todas as transições são disparadas, ou seja, nenhuma transição é morta.

Se as condições 1 e 2 acima são satisfeitas, a Workflow net analisada é *sound*.

Considerando as árvores de prova construídas para os cenários Sc_1 e Sc_2 , a Workflow net mostrada na Figura 2.3 é *sound*, uma vez que as três regras para *soundness* são satisfeitas:

- Apenas um átomo *End* foi produzido em cada cenário.
- Considerando as árvores de prova finalizadas, não há nenhum átomo disponível para consumo nestas árvores de prova.
- Considerando os cenários Sc_1 e Sc_2 da Workflow net analisada, cada transição desta Workflow net aparece em pelo menos um cenário e cada uma destas transições foi

disparada. Assim, é possível afirmar que não há nenhuma transição morta nesta Workflow net.

Considere que deseja-se verificar a propriedade *soundness* para a Workflow net da Figura 5.1.

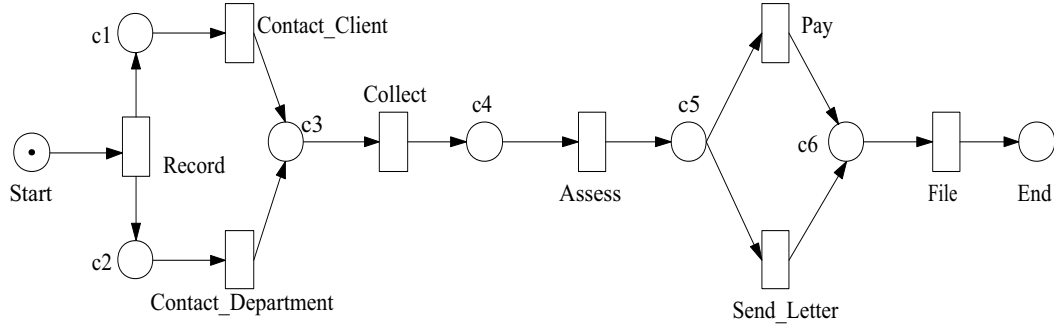


Figura 5.1: Workflow net, com erro de modelagem, para o processo de tratamento de reclamações.

Para tanto, dois sequentes da lógica linear deverão ser provados. O sequente

$$Start, Record, Contact_Client, Contact_Department, Collect, Assess, Pay, File \vdash End$$

precisa ser provado para o cenário Sc_3 , onde a tarefa *Pay* será executada e o sequente

$$Start, Record, Contact_Client, Contact_Department, Collect, Assess, Send_Letter, File \vdash End$$

deverá ser provado para o cenário Sc_4 , onde a tarefa *Send_Letter* será executada, considerando que:

$$Record = t_1 = Start \multimap c1 \otimes c2,$$

$$Contact_Client = t_2 = c1 \multimap c3,$$

$$Contact_Department = t_3 = c2 \multimap c3,$$

$$Collect = t_4 = c3 \multimap c4,$$

$$Assess = t_5 = c4 \multimap c5,$$

$$Pay = t_6 = c5 \multimap c6,$$

$$Send_Letter = t_7 = c5 \multimap c6,$$

$$File = t_8 = c6 \multimap End.$$

A árvore de prova para o cenário Sc_3 se segue:

$$\begin{array}{c}
\frac{c6 \vdash c6 \quad c3, End \vdash End}{\rightarrow_L} \\
\frac{c5 \vdash c5 \quad c3, c6, c6 \multimap End \vdash End}{\rightarrow_L} \\
\frac{c4 \vdash c4 \quad c3, c5, c5 \multimap c6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c3 \vdash c3 \quad c3, c4, c4 \multimap c5, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c2 \vdash c2 \quad c3, c3, c3 \multimap c4, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1 \vdash c1 \quad c2, c3, c2 \multimap c3, t_4, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}{\otimes_L} \\
\frac{Start \vdash Start \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End
\end{array}$$

A árvore de prova para o cenário Sc_4 se segue:

$$\begin{array}{c}
\frac{c6 \vdash c6 \quad c3, End \vdash End}{\rightarrow_L} \\
\frac{c5 \vdash c5 \quad c3, c6, c6 \multimap End \vdash End}{\rightarrow_L} \\
\frac{c4 \vdash c4 \quad c3, c5, c5 \multimap c6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c3 \vdash c3 \quad c3, c4, c4 \multimap c5, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c2 \vdash c2 \quad c3, c3, c3 \multimap c4, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1 \vdash c1 \quad c2, c3, c2 \multimap c3, t_4, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1, c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}{\otimes_L} \\
\frac{Start \vdash Start \quad c1 \otimes c2, c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
Start, Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End
\end{array}$$

Considerando as árvores de prova para os cenários Sc_3 e Sc_4 , a Workflow net mostrada na Figura 5.1 não é *sound*, porque a condição 1b não é satisfeita, isto é, há um átomo $c3$ disponível para consumo nas árvores de prova. Este fato mostra que quando uma ficha é produzida no lugar End há um outro lugar da Workflow net, neste caso $c3$, que não está vazio e, assim, o segundo critério da propriedade *soundness* não é satisfeito.

5.2 Análise Quantitativa: Planejamento de Recursos

O principal objetivo do planejamento de recursos, apresentado neste trabalho, é calcular janelas de datas onde poderá ocorrer a utilização de um recurso (pessoa, equipe, empresa terceirizada etc.) para o tratamento de atividades. Assim, será possível prever a disponibilidade desejada dos recursos envolvidos na execução das tarefas correspondentes.

No caso das Workflow nets analisadas através de árvores de prova canônica da lógica

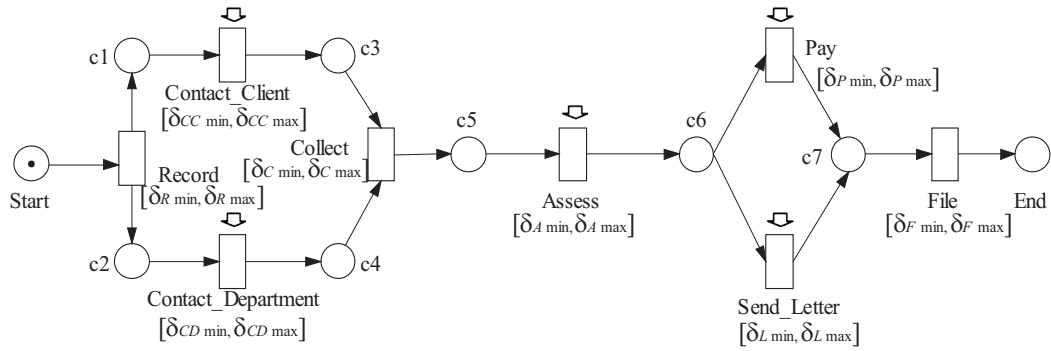


Figura 5.2: t-Time WorkFlow net com datas simbólicas para o processo de tratamento de reclamação e seus acionamentos.

linear, as datas de execução das tarefas poderão ser dadas através de datas simbólicas, ao invés de datas numéricas.

A maior vantagem da utilização de datas simbólicas é que quando estas já estiverem calculadas poderão ser utilizadas diretamente para qualquer caso que será tratado pelo processo de workflow analisado – mesmo considerando casos cujo instante de iniciação e/ou duração global de atividades sejam diferentes. Através das árvores de prova, datas simbólicas para a execução de tarefas podem ser derivadas.

A análise quantitativa também considera todos os possíveis cenários de uma t-Time WorkFlow net. Isto é necessário uma vez que o planejamento de recursos precisa calcular a data de utilização dos recursos associados a cada tarefa presente no processo de workflow analisado.

Como exemplo, considera-se a t-Time WorkFlow net mostrada na Figura 5.2. Para realizar o planejamento de recursos para esta WorkFlow net é necessário construir árvores de prova canônica com cálculo de datas para os cenários Sc_1 e Sc_2 . Os cenários Sc_1 e Sc_2 aqui tratados são os mesmos cenários Sc_1 e Sc_2 definidos na Seção 5.1. Para tanto, considera-se que: $Seq = D_S + d_R + \max(d_{CC}, d_{CD}) + d_C$. As árvores de prova canônica, com cálculo de datas, que correspondem aos cenários Sc_1 e Sc_2 são apresentadas na sequência.

Árvore de prova, com cálculo de datas, para o cenário Sc_1 :

$$\begin{array}{c}
\frac{c7(Seq+d_A+d_P, Seq+d_A+d_P+d_F) \vdash c7 \quad End(Seq+d_A+d_P+d_F, \cdot) \vdash End}{\rightarrow_L} \\
\frac{c6(Seq+d_A, Seq+d_A+d_P) \vdash c6 \quad c7(Seq+d_A+d_P, \cdot), c7 \multimap End \vdash End}{\rightarrow_L} \\
\frac{c5(Seq, Seq+d_A) \vdash c5 \quad c6(Seq+d_A, \cdot), c6 \multimap c7, t_8 \vdash End}{\rightarrow_L} \\
\frac{\frac{c3(D_S+d_R+d_{CC}, Seq) \vdash c3 \quad c4(D_S+d_R+d_{CD}, Seq) \vdash c4}{c3(D_S+d_R+d_{CC}, Seq), c4(D_S+d_R+d_{CD}, Seq) \vdash c3 \otimes c4} \otimes_R \quad c5(Seq, \cdot), c5 \multimap c6, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c2(D_S+d_R, D_S+d_R+d_{CD}) \vdash c2 \quad c3(D_S+d_R+d_{CC}, \cdot), c4(D_S+d_R+d_{CD}, \cdot), c3 \otimes c4 \multimap c5, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1(D_S+d_R, D_S+d_R+d_{CC}) \vdash c1 \quad c2(D_S+d_R, \cdot), c3(D_S+d_R+d_{CC}, \cdot), c2 \multimap c4, t_4, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1(D_S+d_R, \cdot), c2(D_S+d_R, \cdot), c1 \multimap c3, t_3, t_4, t_5, t_6, t_8 \vdash End}{\otimes_L} \\
\frac{Start(D_S, D_S+d_R) \vdash Start \quad c1(D_S+d_R, \cdot) \otimes c2(D_S+d_R, \cdot), t_2, t_3, t_4, t_5, t_6, t_8 \vdash End}{\rightarrow_L} \\
Start(D_S, \cdot), Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_6, t_8 \vdash End
\end{array}$$

Árvore de prova, com cálculo de datas, para o cenário Sc_2 :

$$\begin{array}{c}
\frac{c7(Seq+d_A+d_L, Seq+d_A+d_L+d_F) \vdash c7 \quad End(Seq+d_A+d_L+d_F, \cdot) \vdash End}{\rightarrow_L} \\
\frac{c6(Seq+d_A, Seq+d_A+d_L) \vdash c6 \quad c7(Seq+d_A+d_L, \cdot), c7 \multimap End \vdash End}{\rightarrow_L} \\
\frac{c5(Seq, Seq+d_A) \vdash c5 \quad c6(Seq+d_A, \cdot), c6 \multimap c7, t_8 \vdash End}{\rightarrow_L} \\
\frac{\frac{c3(D_S+d_R+d_{CC}, Seq) \vdash c3 \quad c4(D_S+d_R+d_{CD}, Seq) \vdash c4}{c3(D_S+d_R+d_{CC}, Seq), c4(D_S+d_R+d_{CD}, Seq) \vdash c3 \otimes c4} \otimes_R \quad c5(Seq, \cdot), c5 \multimap c7, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c2(D_S+d_R, D_S+d_R+d_{CD}) \vdash c2 \quad c3(D_S+d_R+d_{CC}, \cdot), c4(D_S+d_R+d_{CD}, \cdot), c3 \otimes c4 \multimap c5, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1(D_S+d_R, D_S+d_R+d_{CC}) \vdash c1 \quad c2(D_S+d_R, \cdot), c3(D_S+d_R+d_{CC}, \cdot), c2 \multimap c4, t_4, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
\frac{c1(D_S+d_R, \cdot), c2(D_S+d_R, \cdot), c1 \multimap c3, t_3, t_4, t_5, t_7, t_8 \vdash End}{\otimes_L} \\
\frac{Start(D_S, D_S+d_R) \vdash Start \quad c1(D_S+d_R, \cdot) \otimes c2(D_S+d_R, \cdot), t_2, t_3, t_4, t_5, t_7, t_8 \vdash End}{\rightarrow_L} \\
Start(D_S, \cdot), Start \multimap c1 \otimes c2, t_2, t_3, t_4, t_5, t_7, t_8 \vdash End
\end{array}$$

Para cada tarefa do tipo usuário modelada na WorkFlow net analisada, deve-se extrair as datas de produção, D_P , e consumo, D_C , do átomo que representa a pré-condição da transição correspondente a esta tarefa. Quando há mais de uma pré-condição associada à transição, considera-se a data máxima das produções dos átomos correspondentes a estas pré-condições. A data de produção deste átomo, D_P , corresponde ao início da execução da tarefa associada à transição e a data de consumo, D_C , corresponde ao término da execução da mesma. Assim, é gerado um intervalo $[D_P, D_C]$ de datas onde o recurso que executará a referida tarefa deverá ser disponibilizado para realizá-la.

Uma vez que as datas de produção e consumo são dependentes de durações de sensibilização d_i , cujo valor pertence a um intervalo de tempo $\Delta_i = [\delta_{imin}, \delta_{imax}]$, pode-se considerar vários intervalos possíveis de execução das tarefas, de acordo com um planejamento estratégico. Por exemplo, o intervalo de execução $I_{Exec} = [D_{Pmin}, D_{Cmax}]$ considera que a alocação do recurso para a execução da tarefa poderá ocorrer entre o início ao mais

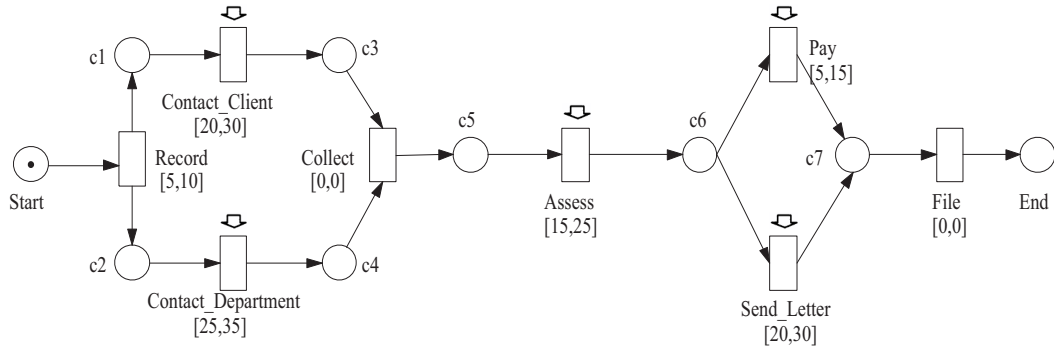


Figura 5.3: t-Time WorkFlow net com datas numéricas para o processo de tratamento de reclamação e seus acionamentos.

cedo e o término ao mais tarde da atividade considerada. Este intervalo de datas é o mais flexível, no sentido de que ele considera a janela de tempo de utilização do recurso a mais extensa possível.

De acordo com a disponibilidade do recurso envolvido, poderão ser consideradas na estratégia de escalonamento de atividades do processo de workflow outras datas fornecidas pela árvore de prova, como $D_{P_{max}}$ (data de início ao mais tarde da tarefa) ou $D_{C_{min}}$ (data de término ao mais cedo da tarefa).

A informação sobre quando o processo global finalizará, para um dado caso, é dado pela data de produção do átomo *End*. Assim, pode-se calcular a data ao mais cedo de finalização do processo, dada pela data mínima $D_{P_{min}}$ de produção deste átomo e a data ao mais tarde de finalização do processo, dada pela data máxima $D_{P_{max}}$ de produção do átomo *End*.

A Tabela 5.1 mostra os intervalos de datas simbólicas de execução para as tarefas que são do tipo usuário, dos cenários Sc_1 e Sc_2 , considerando o intervalo de execução $I_{Exec} = [D_{P_{min}}, D_{C_{max}}]$.

Os intervalos de datas calculados poderão ser utilizados por qualquer caso tratado pelo processo de workflow modelado pela t-Time WorkFlow net da Figura 5.2.

A título de exemplificação, considera-se o processo de tratamento de reclamações mapeado na t-Time WorkFlow net mostrada na Figura 5.3. A única diferença entre as Figuras 5.2 e 5.3 é que a primeira associa intervalos de tempo simbólico às transições enquanto que a segunda associa às transições intervalos de tempo numérico. Assim, considerando os intervalos de tempo da t-Time WorkFlow net mostrada na Figura 5.3 e considerando que o processo a ser tratado inicia-se na data 0, ou seja, $D_S = 0$, os intervalos de datas para a execução das tarefas podem ser calculados apenas substituindo as datas simbólicas presentes na Tabela 5.1 pelas datas numéricas associadas às transições da t-Time WorkFlow net da Figura 5.3. A Tabela 5.2 mostra o resultado deste processo. Por exemplo, o recurso utilizado para executar a tarefa *Send_Letter* do cenário Sc_2 deverá ser alocado ao mais cedo na data 40 e deverá ser liberado ao mais tarde na data 100.

Assim, o recurso que executará esta tarefa poderá ser alocado neste intervalo de datas.

Transição	Intervalos de data Sc1 e Sc2
$t_2=c1 \rightarrow c3$	$[D_S + \delta_{Rmin}, D_S + \delta_{Rmax} + \delta_{CCmax}]$
$t_3=c2 \rightarrow c4$	$[D_S + \delta_{Rmin}, D_S + \delta_{Rmax} + \delta_{CDmax}]$
$t_5=c5 \rightarrow c6$	$[D_S + \delta_{Rmin} + \max(\delta_{CCmin}, \delta_{CDmin}) + \delta_{Cmin},$ $D_S + \delta_{Rmax} + \max(\delta_{CCmax}, \delta_{CDmax}) + \delta_{Cmax} + \delta_{Amax}]$
Transição	Intervalos de data Sc1
$t_6=c6 \rightarrow c7$	$[D_S + \delta_{Rmin} + \max(\delta_{CCmin}, \delta_{CDmin}) + \delta_{Cmin} + \delta_{Amin},$ $D_S + \delta_{Rmax} + \max(\delta_{CCmax}, \delta_{CDmax}) + \delta_{Cmax} + \delta_{Amax} + \delta_{Pmax}]$
Transição	Intervalos de data Sc2
$t_7=c6 \rightarrow c7$	$[D_S + \delta_{Rmin} + \max(\delta_{CCmin}, \delta_{CDmin}) + \delta_{Cmin} + \delta_{Amin},$ $D_S + \delta_{Rmax} + \max(\delta_{CCmax}, \delta_{CDmax}) + \delta_{Cmax} + \delta_{Amax} + \delta_{Lmax}]$

Tabela 5.1: Intervalos de datas simbólicas para execução de tarefas do tipo usuário dos cenários Sc_1 e Sc_2 .

Tarefa	Intervalos de data Sc1	Intervalos de data Sc2
<i>Contact_Client</i>	[5,40]	[5,40]
<i>Contact_Department</i>	[5,45]	[5,45]
<i>Assess</i>	[25,70]	[25,70]
<i>Pay</i>	[40,85]	—
<i>Send_Letter</i>	—	[40,100]

Tabela 5.2: Intervalos de datas numéricas para a execução de tarefas do tipo usuário dos cenários Sc_1 e Sc_2 .

5.3 Considerações Finais do Capítulo

Este capítulo apresentou as abordagens para a análise qualitativa e quantitativa de Workflow nets utilizando como exemplo ilustrativo o processo de tratamento de reclamações apresentado em [van der Aalst e van Hee 2004].

Capítulo 6

Verificação de Especificações de Workflow em UML usando Transformações Automáticas para WorkFlow nets

O objetivo deste capítulo é mostrar como um Diagrama de Atividades da UML, que modela um processo de workflow, pode ser transformado em uma WorkFlow net correspondente, para a qual a propriedade *soundness* deverá ser verificada, através das árvores de prova canônica da lógica linear. Caso esta propriedade não se verifique, as árvores de prova canônica da lógica linear construídas para provar esta propriedade poderão ser analisadas de forma a indicar pontos críticos no Diagrama de Atividades da UML fonte. Para tanto, as regras para a transformação automática de Diagramas de Atividades da UML em WorkFlow nets são mostradas na Seção 6.1. A Seção 6.2 mostra como encontrar e corrigir os pontos críticos nos Diagramas de Atividade da UML através da análise das árvores de prova canônica da lógica linear construídas para provar a propriedade *soundness* para a WorkFlow net correspondente.

6.1 Transformações Automáticas de Diagramas de Atividades da UML em WorkFlow nets

A maior vantagem de prover a transformação de um Diagrama de Atividades da UML em uma rede de Petri é possibilitar a transformação de um modelo semi-formal em um modelo formal, para o qual boas propriedades podem ser verificadas, como é o caso da propriedade *soundness* definida para as WorkFlow nets.

Desta forma, vários esforços vem sendo realizados no âmbito de prover tal transformação. Uma abordagem para analisar processos de workflow usando transformações entre

modelos é apresentada em [Meena et al. 2005] por exemplo. Os autores propõem uma transformação de um Diagrama de Atividades da UML em uma rede de Petri, mas apresentam apenas um exemplo de como se dá tal transformação, sem nenhuma formalização das regras de transformação utilizadas. Em [Gehrke et al. 1998], os autores propõem a transformação de um Diagrama de Atividades da UML em uma rede de Petri, apresentando apenas alguns esquemas de transformação sem apresentar as regras genéricas para tanto. Em [Tričković 2000], uma formalização de um Diagrama de Atividades da UML utilizando redes de Petri é proposta. Entretanto, o autor apenas mostra alguns exemplos de transformação, sendo que as regras para tal transformação também não são apresentadas. Em [Störrle et al. 2005], é proposta uma formalização da semântica dos Diagramas de Atividades da UML, mas as regras de transformação apresentadas são insuficientes para atingir uma transformação completa, uma vez que duas regras necessárias para prover a transformação não são apresentadas no trabalho de [Störrle et al. 2005].

O presente trabalho propõe a transformação de um Diagrama de Atividades da UML em uma Workflow net, baseada nos seus respectivos metamodelos que foram apresentados nas Seções 4.2.1 e 2.2.4. No entanto, algumas restrições são impostas ao Diagrama de Atividades da UML a ser transformado: apenas um *Initial Node* e um *Final Node* podem aparecer no diagrama, todas as atividades (Action) precisam ter, pelo menos, uma aresta (Control Flow) saindo delas e de forma que esta aresta tenha como alvo uma atividade diferente daquela que é a atividade predecessora da atividade da qual ela sai. Sem estas restrições e utilizando a transformação proposta neste trabalho, poderia-se obter por exemplo uma transição sem arcos de saída (*Transition to Place*) ou mais de um lugar *Start* ou *End*, o que não é possível acontecer em uma Workflow net.

6.1.1 Regras para a transformação de um Diagrama de Atividades da UML em uma Workflow net

Algumas das regras aqui propostas para a transformação de um Diagrama de Atividades da UML em uma Workflow net foram anteriormente apresentadas em [Störrle et al. 2005]. Mas como somente as regras apresentadas em [Störrle et al. 2005] são insuficientes para alcançar a transformação proposta, duas novas regras são apresentadas neste trabalho. Tais regras estão destacadas na Figura 6.1 através de um retângulo pontilhado. Com a adição dessas duas novas regras, a transformação proposta é atingida. A Figura 6.1 apresenta as regras de transformação intuitivamente.

De acordo com as regras de transformação mostradas na Figura 6.1:

- o elemento de modelagem *Action* do Diagrama de Atividades da UML será transformado no elemento de modelagem *Transition* de uma Workflow net;
- os elementos de modelagem *ForkNode* ou *JoinNode* do Diagrama de Atividades da UML serão transformados no elemento de modelagem *Transition* da Workflow net;

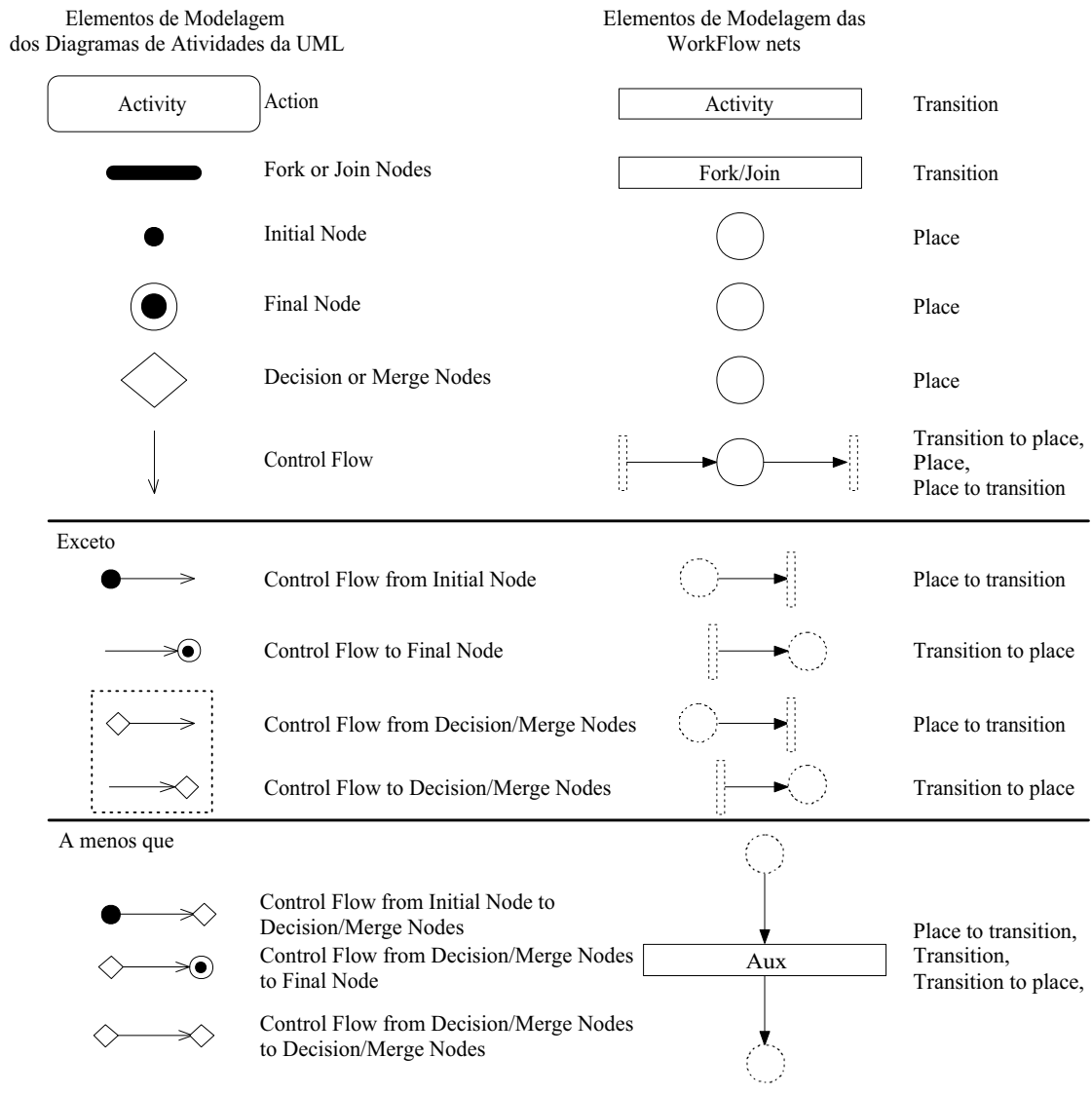


Figura 6.1: Regras de transformação.

- o elemento de modelagem *InitialNode* do Diagrama de Atividades da UML será transformado no elemento de modelagem *Place* da WorkFlow net;
- o elemento de modelagem *FinalNode* do Diagrama de Atividades da UML será transformado no elemento de modelagem *Place* da WorkFlow net;
- os elementos de modelagem *DecisionNode* ou *MergeNode* do Diagrama de Atividades da UML serão transformados no elemento de modelagem *Place* da WorkFlow net;
- o elemento de modelagem *ControlFlow* do Diagrama de Atividades da UML será transformado em um conjunto de elementos de modelagem *TransitionToPlace*, *Place* e *PlaceToTransition* da WorkFlow net, exceto nos casos em que:
 - o elemento *ControlFlow* tiver como elemento anterior um elemento *InitialNode* e neste caso será transformado em um elemento de modelagem *PlaceToTransition* da WorkFlow net;

- o elemento *ControlFlow* tiver como elemento posterior um elemento *FinalNode* e neste caso será transformado em um elemento de modelagem *TransitionToPlace* da Workflow net;
- o elemento *ControlFlow* tiver como elemento anterior um elemento *DecisionNode* ou *MergeNode* e neste caso será transformado em um elemento de modelagem *PlaceToTransition* da Workflow net;
- o elemento *ControlFlow* tiver como elemento posterior um elemento *DecisionNode* ou *MergeNode* e neste caso será transformado em um elemento de modelagem *TransitionToPlace* da Workflow net;

A menos que:

- * o elemento *ControlFlow* tiver como elemento anterior um elemento *InitialNode* e como elemento posterior um elemento *DecisionNode* ou *MergeNode*;
- * o elemento *ControlFlow* tiver como elemento anterior um elemento *DecisionNode* ou *MergeNode* e como elemento posterior um elemento *FinalNode*;
- * o elemento *ControlFlow* tiver como elemento anterior um elemento *DecisionNode* ou *MergeNode* e como elemento posterior um elemento *DecisionNode* ou *MergeNode*;

sendo que nestes três casos o elemento *ControlFlow* será transformado em um conjunto de elementos *PlaceToTransition*, *Transition* e *TransitionToPlace* da Workflow net.

O metamodelo das Workflow nets e o metamodelo dos Diagramas de Atividades da UML são mostrados nas Figuras 2.5 e 4.3, respectivamente, e serão utilizados como base na transformação entre os respectivos modelos. Portanto, é interessante rever os metamodelos citados para que se possa melhor entender as regras de transformação.

O código ATL para o mapeamento de Diagramas de Atividades da UML em Workflow nets é mostrado no Apêndice.

Considere o Diagrama de Atividades da UML mostrado na Figura 6.2(a). Suponha que deseja-se transformar este diagrama em uma Workflow net correspondente. Para tanto, as regras *InitialNode2Place*, *ControlFlowFromInitialNode*, *Action2Transition*, *ControlFlow*, *ControlFlowToFinalNode* e *FinalNode2Place* serão utilizadas. A regra *InitialNode2Place* irá transformar o *InitialNode* deste diagrama, que tem atributo “*name = Start*”, em um *Place*, também com atributo “*name = Start*”, da Workflow net correspondente. A regra *ControlFlowFromInitialNode* será utilizada para transformar o elemento de modelagem *ControlFlow* que tem como elemento anterior o *InitialNode* deste Diagrama de Atividades em um elemento de modelagem *PlaceToTransition* da Workflow net correspondente, de forma que as associações “*from*” e “*to*” deste elemento de modelagem *PlaceToTransition* receberão os valores das associações “*source*” e “*target*” do elemento *ControlFlow*, respectivamente. A regra *Action2Transition* será aplicada na primeira vez para transformar

o elemento *Action* que tem valor do atributo “*name = Atividade1*” deste Diagrama de Atividades em um elemento *Transition* da WorkFlow net correspondente, de forma que o atributo “*name*” deste elemento *Transition* receberá o atributo “*name*” do elemento *Action*. Na sua segunda aplicação, a regra *Action2Transition* transformará desta mesma maneira um elemento *Action* com atributo “*name = Atividade2*” em um elemento *Transition* com atributo “*name=Atividade2*”. A regra *ControlFlow* será utilizada na transformação do elemento *ControlFlow* que tem como elemento anterior o elemento *Action* com atributo “*name = Atividade1*” e como elemento posterior o elemento *Action* com atributo “*name = Atividade2*” em três elementos de modelagem da WorkFlow net correspondente:

- um elemento *TransitionToPlace*, que receberá como valor da associação “*from*” o valor da associação “*source*” do elemento *ControlFlow* e para a associação “*to*” o elemento de modelagem *Place* que será criado nesta mesma transformação (veja na regra em ATL *ControlFlow* que esta atribuição é feita após a criação deste elemento *Place*);
- um elemento *PlaceToTransition*, que receberá como valor da associação “*to*” o valor da associação “*target*” do elemento *ControlFlow* e para a associação “*from*” o elemento de modelagem *Place* que será criado nesta mesma transformação (veja na regra em ATL *ControlFlow* que esta atribuição é feita após a criação deste elemento *Place*);
- um elemento *Place*, que receberá como valor da associação “*incomingEdge*” o elemento *TransitionToPlace* criado nesta transformação e como valor da associação “*outgoingEdge*” o elemento *PlaceToTransition* também criado nesta mesma transformação.

A regra *ControlFlowToFinalNode* será utilizada para transformar o elemento de modelagem *ControlFlow* que tem como elemento posterior o elemento *FinalNode* deste Diagrama de Atividades em um elemento *TransitionToPlace* da WorkFlow net correspondente, que terá como valor das associações “*from*” e “*to*” o valor das associações “*source*” e “*target*” do elemento *ControlFlow*. A regra *FinalNode2Place* será aplicada para transformar o elemento de modelagem *FinalNode* deste Diagrama de Atividades em um elemento *Place* da WorkFlow net correspondente, de forma que o valor do atributo “*name*” deste elemento receberá o valor do atributo “*name*” do elemento *FinalNode*. A WorkFlow net obtida através desta transformação é apresentada na Figura 6.2(b).

O processo de tratamento de reclamações descrito na Seção 2.2.1 também é utilizado para ilustrar o mapeamento proposto. A Figura 6.3(a) mostra o Diagrama de Atividades da UML que modela esse processo de workflow. Na Figura 6.3(b) é mostrada a WorkFlow net correspondente ao Diagrama de Atividades da UML mostrado na Figura 6.3(a).

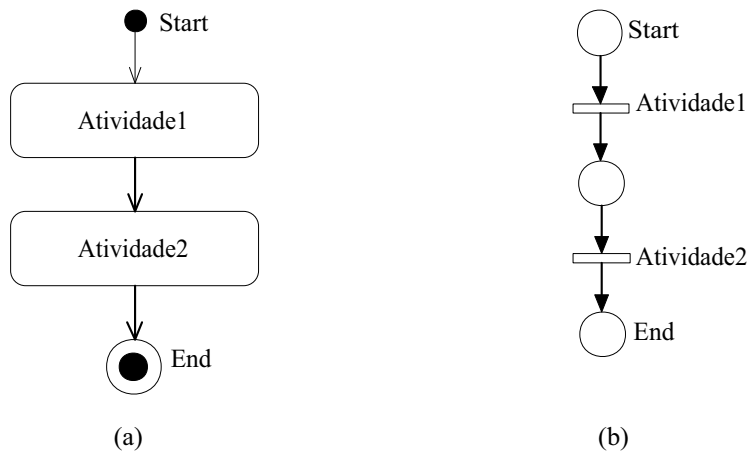


Figura 6.2: Exemplo de transformação.

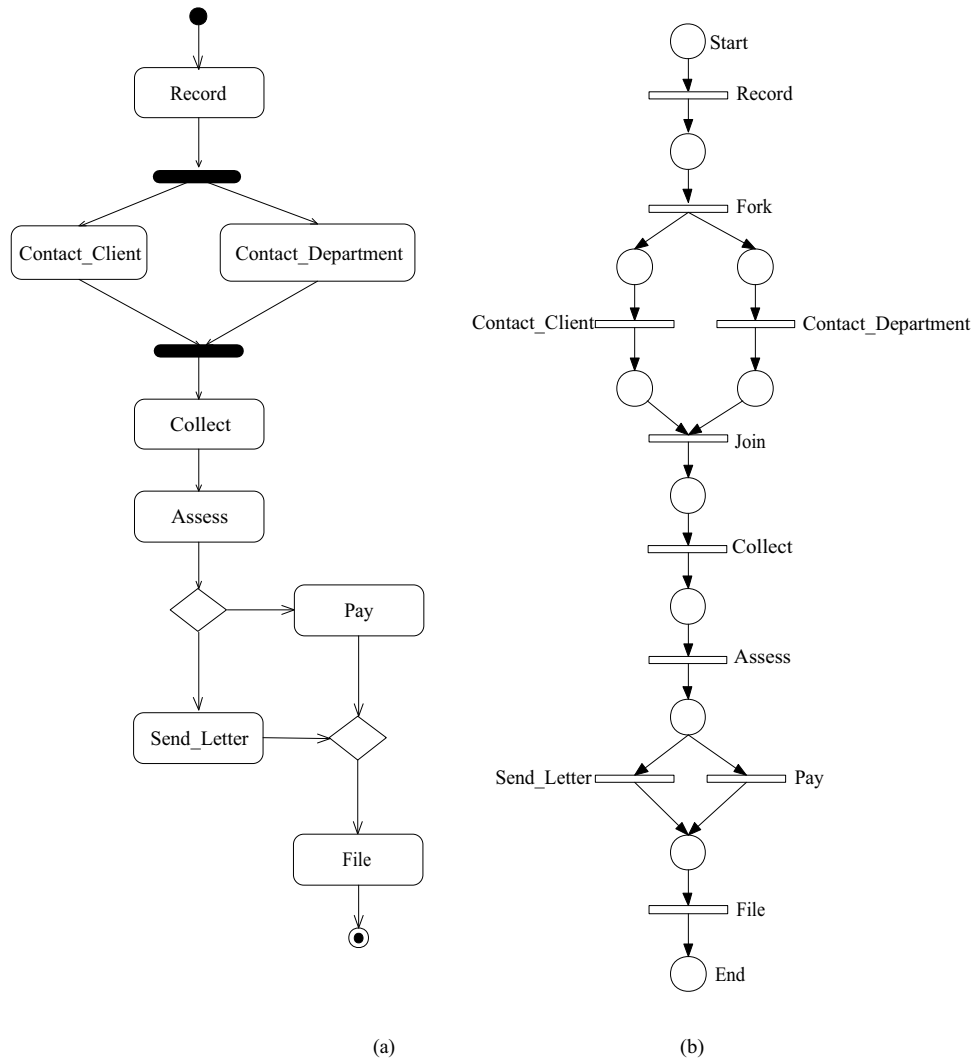


Figura 6.3: O processo de tratamento de reclamações. (a) Processo de workflow modelado através de um Diagrama de Atividades da UML. (b) Processo de workflow modelado através de uma WorkFlow net.

6.2 Encontrando e Corrigindo Pontos Críticos em Diagramas de Atividades da UML utilizando Árvores de Prova Canônica da Lógica Linear

No contexto deste trabalho, um ponto crítico em um Diagrama de Atividades da UML corresponde a uma região deste diagrama que possua um erro de modelagem no contexto dos processos de workflow. Vale destacar que este ponto crítico não corresponde a um erro de modelagem no que diz respeito à modelagem UML genérica, sendo que o Diagrama de Atividades da UML pode estar sintaticamente correto neste contexto mas não no contexto de processos de workflow.

Alguns pontos críticos em Diagramas de Atividades, que modelam processos de workflow, podem ser encontrados através da análise das árvores de prova canônica da lógica linear, contruídas para provar a propriedade *soundness* de sua WorkFlow net correspondente.

Para ilustrar esta abordagem é apresentada uma transformação de um Diagrama de Atividades da UML em uma WorkFlow net. Depois disso, é realizada a construção das árvores de prova para tentar provar que a WorkFlow net obtida na transformação satisfaz o critério de correteza *soundness*, o que não acontece no caso do exemplo apresentado. A partir daí, é realizada a análise dos pontos críticos, baseada nas árvores de prova da lógica linear, e a correção do Diagrama de Atividades da UML original.

O processo de workflow analisado é apresentado em [OMG 2008] e é baseado na seleção padrão de peças de um processo de projeto de uma companhia de transporte aéreo: “O comportamento da atividade *Expert Part Search* pode resultar em uma peça encontrada ou não. Quando a peça não foi encontrada o processo requer a atividade *Assign Standards Engineer*. Por último, a invocação da atividade *Schedule Part Mod Workflow* resulta em atividades globais que são passadas para as invocações de escalonamento e execução subsequentes, ou seja, *Execute Part Mod Workflow* e *Research Production Possibility*. A Figura 6.4 mostra o Diagrama de Atividades da UML que modela esse processo. A Figura 6.5 mostra a WorkFlow net obtida aplicando as devidas transformações ao Diagrama de Atividades da UML da Figura 6.4.

Há quatro cenários distintos na WorkFlow net da Figura 6.5: o primeiro cenário, Sc_1 , onde a tarefa *Aux1* será executada (disparando a transição *Aux1*), o segundo cenário, Sc_2 , onde a tarefa *Aux2* será executada, o terceiro cenário, Sc_3 , onde a tarefa *Join* será executada e o quarto cenário, Sc_4 , onde a tarefa *Aux3* será executada. Então, para esta WorkFlow net, quatro sequentes da lógica linear deverão ser provados. Para o cenário Sc_1 , é necessário provar o sequente $Start, t_1, t_{11} \vdash End$, para o cenário Sc_2 o sequente $Start, t_1, t_2, t_3, t_4, t_5, t_6, t_{12} \vdash End$, para o cenário Sc_3 o sequente $Start, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \vdash End$ e para o cenário Sc_4 o sequente $Start, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{13} \vdash End$.

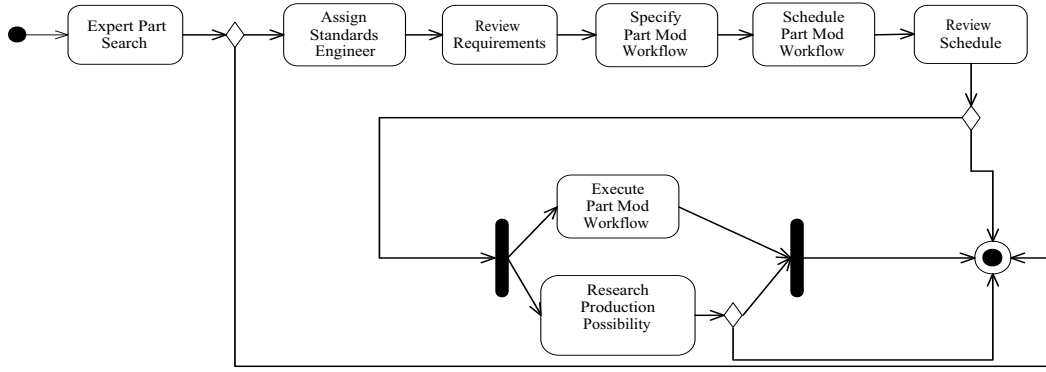


Figura 6.4: Diagrama de Atividades da UML que modela o processo de seleção de peças.

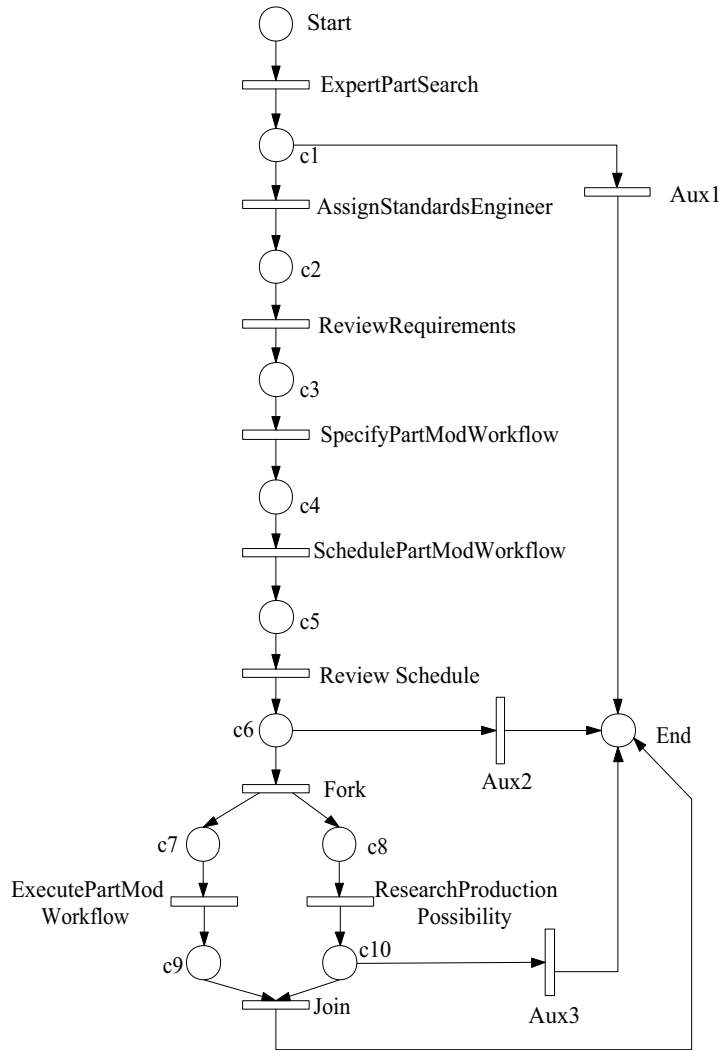


Figura 6.5: WorkFlow net que modela o processo de seleção de peças.

precisam ser provados, considerando que:

$$t_1 = \text{ExpertPartSearch} = \text{Start} \multimap c1,$$

$$t_2 = \text{AssignStandardsEngineer} = c1 \multimap c2,$$

$$t_3 = \text{ReviewRequirements} = c2 \multimap c3,$$

$$t_4 = \text{SpecifyPartModWorkflow} = c3 \multimap c4,$$

$$\begin{aligned}
 t_5 &= \text{SchedulePartModWorkflow} = c4 \multimap c5, \\
 t_6 &= \text{ReviewSchedule} = c5 \multimap c6, \\
 t_7 &= \text{Fork} = c6 \multimap c7 \otimes c8, \\
 t_8 &= \text{ExecutePartModWorkflow} = c7 \multimap c9, \\
 t_9 &= \text{ResearchProductionPossibility} = c8 \multimap c10, \\
 t_{10} &= \text{Join} = c9 \otimes c10 \multimap \text{End}, \\
 t_{11} &= \text{Aux1} = c1 \multimap \text{End}, \\
 t_{12} &= \text{Aux2} = c6 \multimap \text{End}, \\
 t_{13} &= \text{Aux3} = c10 \multimap \text{End}.
 \end{aligned}$$

Na sequência são mostradas as árvores de prova para os cenários Sc_1 , Sc_2 , Sc_3 e Sc_4 .

Árvore de prova para o cenário Sc_1 :

$$\begin{array}{c}
 \frac{c1 \vdash c1 \quad \text{End} \vdash \text{End}}{\multimap_L} \\
 \frac{\text{Start} \vdash \text{Start} \quad \frac{c1, c1 \multimap \text{End} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \text{Start}, \text{Start} \multimap c1, c1 \multimap \text{End} \vdash \text{End}
 \end{array}$$

Árvore de prova para o cenário Sc_2 :

$$\begin{array}{c}
 \frac{c6 \vdash c6 \quad \text{End} \vdash \text{End}}{\multimap_L} \\
 \frac{c5 \vdash c5 \quad \frac{c6, c6 \multimap \text{End} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \frac{c4 \vdash c4 \quad \frac{c5, c5 \multimap c6, t_{12} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \frac{c3 \vdash c3 \quad \frac{c4, c4 \multimap c5, t_6, t_{12} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \frac{c2 \vdash c2 \quad \frac{c3, c3 \multimap c4, t_5, t_6, t_{12} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \frac{c1 \vdash c1 \quad \frac{c2, c2 \multimap c3, t_4, t_5, t_6, t_{12} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \frac{\text{Start} \vdash \text{Start} \quad \frac{c1, c1 \multimap c2, t_3, t_4, t_5, t_6, t_{12} \vdash \text{End}}{\multimap_L}}{\multimap_L} \\
 \text{Start}, \text{Start} \multimap c1, t_2, t_3, t_4, t_5, t_6, t_{12} \vdash \text{End}
 \end{array}$$

Árvore de prova para o cenário Sc_3 :

$$\begin{array}{c}
 \frac{\frac{c9 \vdash c9 \quad c10 \vdash c10}{c9, c10 \vdash c9 \otimes c10} \otimes_R \quad End \vdash End}{c8 \vdash c8 \quad c9, c10, c9 \otimes c10 \multimap End \vdash End} \multimap_L \\
 \frac{c7 \vdash c7 \quad c8, c9, c8 \multimap c10, t_{10} \vdash End}{c7, c8, c7 \multimap c9, t_9, t_{10} \vdash End} \otimes_L \\
 \frac{c6 \vdash c6 \quad c7 \otimes c8, t_8, t_9, t_{10} \vdash End}{c5 \vdash c5 \quad c6, c6 \multimap c7 \otimes c8, t_8, t_9, t_{10} \vdash End} \multimap_L \\
 \frac{c4 \vdash c4 \quad c5, c5 \multimap c6, t_7, t_8, t_9, t_{10} \vdash End}{c3 \vdash c3 \quad c4, c4 \multimap c5, t_6, t_7, t_8, t_9, t_{10} \vdash End} \multimap_L \\
 \frac{c2 \vdash c2 \quad c3, c3 \multimap c4, t_5, t_6, t_7, t_8, t_9, t_{10} \vdash End}{c1 \vdash c1 \quad c2, c2 \multimap c3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \vdash End} \multimap_L \\
 \frac{Start \vdash Start \quad c1, c1 \multimap c2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \vdash End}{Start, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \vdash End} \multimap_L
 \end{array}$$

Árvore de prova para o cenário Sc_4 :

$$\begin{array}{c}
 \frac{c10 \vdash c10 \quad c9, End \vdash End}{c8 \vdash c8 \quad c9, c10, c10 \multimap End \vdash End} \multimap_L \\
 \frac{c7 \vdash c7 \quad c8, c9, c8 \multimap c10, t_{13} \vdash End}{c7, c8, c7 \multimap c9, t_9, t_{13} \vdash End} \otimes_L \\
 \frac{c6 \vdash c6 \quad c7 \otimes c8, t_8, t_9, t_{13} \vdash End}{c5 \vdash c5 \quad c6, c6 \multimap c7 \otimes c8, t_8, t_9, t_{13} \vdash End} \multimap_L \\
 \frac{c4 \vdash c4 \quad c5, c5 \multimap c6, t_7, t_8, t_9, t_{13} \vdash End}{c3 \vdash c3 \quad c4, c4 \multimap c5, t_6, t_7, t_8, t_9, t_{13} \vdash End} \multimap_L \\
 \frac{c2 \vdash c2 \quad c3, c3 \multimap c4, t_5, t_6, t_7, t_8, t_9, t_{13} \vdash End}{c1 \vdash c1 \quad c2, c2 \multimap c3, t_4, t_5, t_6, t_7, t_8, t_9, t_{13} \vdash End} \multimap_L \\
 \frac{Start \vdash Start \quad c1, c1 \multimap c2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{13} \vdash End}{Start, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{13} \vdash End} \multimap_L
 \end{array}$$

Algumas análises podem ser realizadas baseando-se nas árvores de prova da lógica linear, construídas para a prova dos sequentes da lógica linear que definem os cenários de uma Workflow net. Por exemplo, se uma dada transição, representada por uma fórmula da lógica linear, não foi disparada em nenhum cenário, significa que esta é uma transição morta presente na Workflow net, e a atividade representada por esta transição não é necessária ao processo de workflow, uma vez que a mesma jamais será executada. Assim,

algumas alterações de remoção devem ser realizadas no Diagrama de Atividades da UML para tornar a Workflow net correspondente *sound*.

Uma outra situação é encontrada quando, após a finalização da construção de uma dada árvore de prova, ainda existe um átomo disponível para consumo nesta árvore de prova: isso indica que há um ponto crítico na região do lugar representado pelo átomo disponível para consumo. Por exemplo, analisando a árvore de prova para o cenário Sc_4 , pode-se notar que há um átomo c_9 disponível para consumo nesta árvore de prova, mesmo depois de sua finalização. Isso indica que na região do lugar c_9 da Workflow net mostrada na Figura 6.5 há algum erro de modelagem. Assim, o analista pode estudar e corrigir a região correspondente a esta no Diagrama de Atividades da UML, preservando a semântica necessária. A Figura 6.6 mostra um Diagrama de Atividades da UML com uma possível correção para este exemplo. Esta correção determina que o processo pode finalizar apenas se ambas as atividades *Research Production Possibility* e *Execute Part Mod Workflow* já estiverem finalizadas ou se a atividade *Research Production Possibility* já estiver finalizada e a atividade *Execute Part Mod Workflow* ainda não tiver se iniciado. Depois da correção, o Diagrama de Atividades da UML atualizado pode ser transformado em uma nova Workflow net, para a qual deverá ser testada a propriedade *soundness*, através da construção das árvores de prova para cada cenário da mesma. Assim, considerando ainda o exemplo de correção, a Figura 6.7 mostra a Workflow net correspondente ao Diagrama de Atividades da UML corrigido. Através da construção das árvores de prova canônica da lógica linear, pode-se observar que esta Workflow net é *sound*.

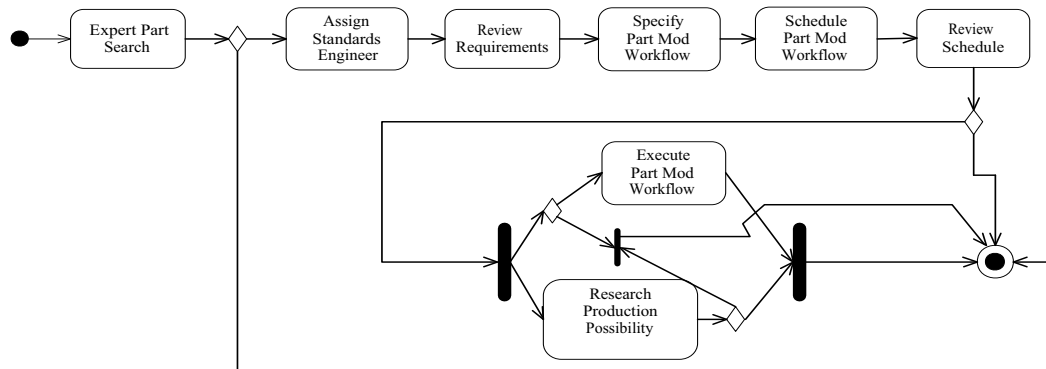


Figura 6.6: Diagrama de Atividades da UML que modela o processo de seleção de peças corrigido.

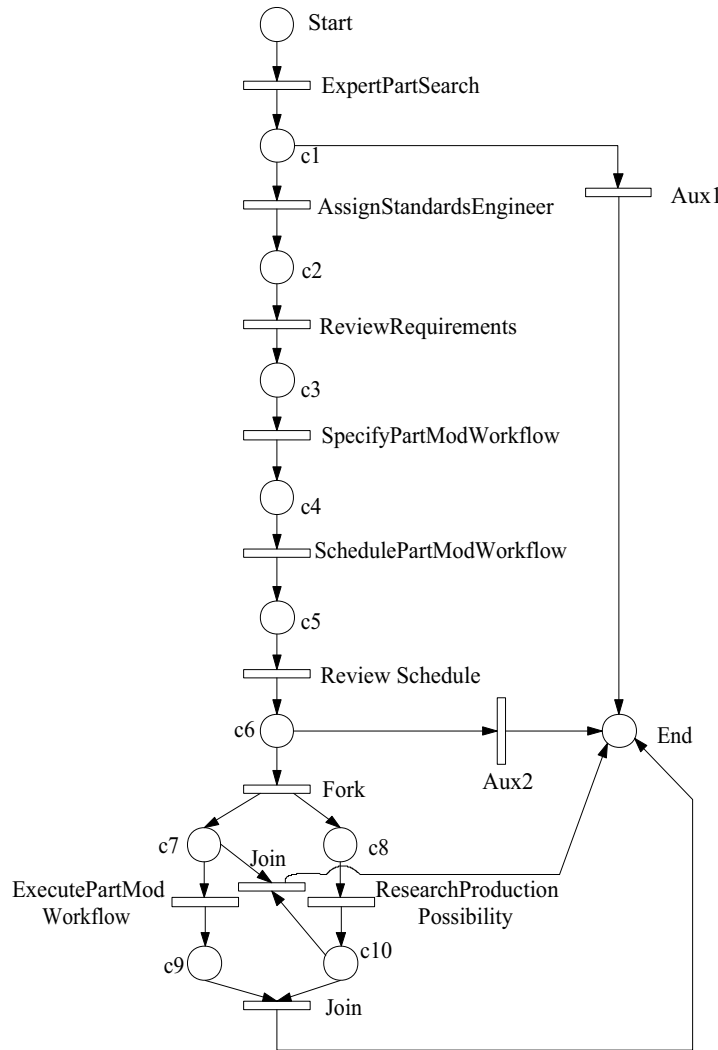


Figura 6.7: WorkFlow net que modela o processo de seleção de peças corrigido.

6.3 Considerações Finais do Capítulo

Este capítulo mostrou como um Diagrama de Atividades da UML, que modela um processo de workflow, pode ser transformado em uma WorkFlow net correspondente e também como encontrar e corrigir erros no Diagrama de Atividades da UML analisando as árvores de prova canônica da lógica linear construídas para provar a propriedade *soundness* para esta WorkFlow net.

Capítulo 7

Estudo de Caso

O presente capítulo tem por objetivo analisar qualitativamente e quantitativamente uma iteração do processo de Engenharia de Software denominado RUP (*Rational Unified Process*). Para tanto, é realizada uma introdução ao RUP e um Diagrama de Atividades da UML que especifica as atividades deste processo é apresentado. Este diagrama é então transformado em uma WorkFlow net correspondente, para a qual serão aplicadas as análises qualitativa e quantitativa propostas neste trabalho.

7.1 Análise Qualitativa e Quantitativa de uma Iteração do RUP

7.1.1 RUP - *Rational Unified Process*

O RUP (*Rational Unified Process*) é um processo de Engenharia de Software [Kruchten 2003] que fornece uma abordagem disciplinada para assumir tarefas e responsabilidades dentro de uma organização de desenvolvimento e tem como principal objetivo assegurar uma produção de software de alta qualidade que satisfaça as necessidades de seus usuários finais dentro de prazo e orçamento previsíveis.

A Figura 7.1 mostra a arquitetura global do RUP. Deve-se observar que o processo tem duas dimensões [Kruchten 2003]:

- O eixo horizontal representa o tempo e mostra como os aspectos do ciclo de vida do processo se desdobram.
- O eixo vertical representa os fluxos essenciais do processo.

A primeira dimensão representa os aspecto dinâmico do processo e é expresso em termos de ciclos, fases, iterações e marcos. Já a segunda dimensão representa o aspecto estático do processo e é descrito em termos de componentes de processo, atividades, fluxos, artefatos e trabalhadores.

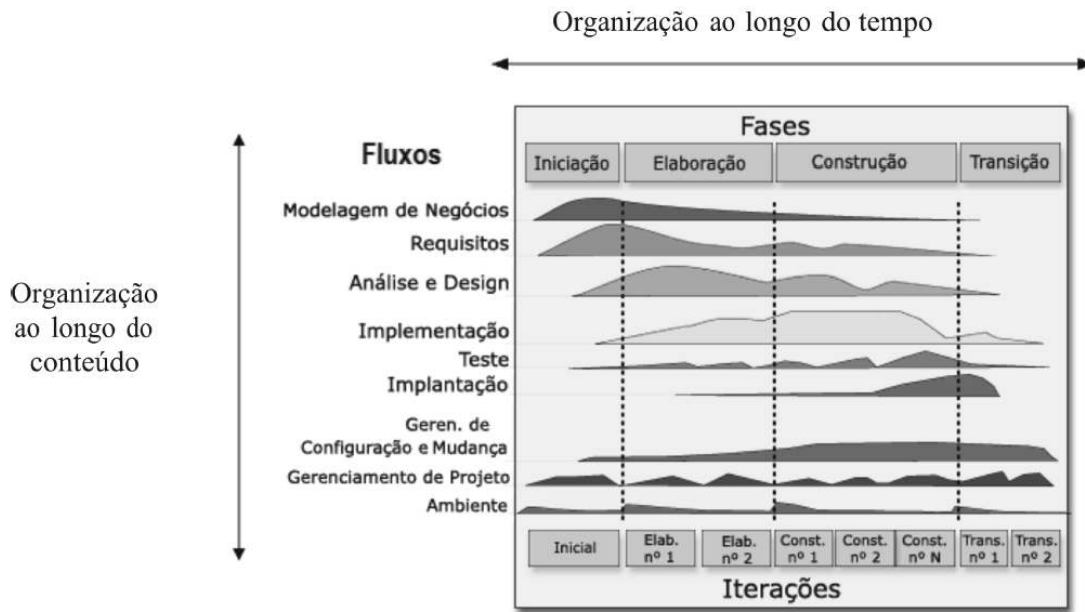


Figura 7.1: Arquitetura global do RUP.

Uma mera enumeração de todas as atividades não constitui totalmente um processo. Desta forma, é necessário descrever as sequências significantes de atividades que produzirão um resultado valioso. Um fluxo é uma sequência de atividades que produz um resultado de valor observável [Kruchten 2003].

No RUP há nove fluxos centrados no processo, sendo que estes são divididos em seis fluxos centrados na engenharia e três de suporte.

Os fluxos de engenharia são:

- Fluxo de modelagem de negócios.
- Fluxo de requisitos.
- Fluxo de análise e design.
- Fluxo de implementação.
- Fluxo de teste.
- Fluxo de implantação.

Os fluxos centrados no suporte são:

- Fluxo de gerenciamento de projeto.
- Fluxo de gerenciamento de mudança e configuração.
- Fluxo de ambiente.

Embora os nomes dos seis fluxos centrados na engenharia possam evocar as fases sequenciais de um processo em cascata tradicional, as fases de um processo iterativo, como proposto pelo RUP, são diferentes, uma vez que estes fluxos são revisitados novamente e

novamente ao longo do ciclo de vida. Assim, a ênfase nas várias atividades muda de uma iteração para a próxima e de uma fase para a seguinte. O fluxo completo de um projeto intercala estes nove fluxos e os repete com várias ênfases e níveis de intensidade a cada iteração [Kruchten 2003].

Em um processo iterativo, recomenda-se que o desenvolvimento esteja baseado em dois tipos de planos:

- Um plano grosseiro: plano de fase.
- Uma série de planos refinados: os planos de iteração, sendo que há um plano para cada iteração.

O plano de iteração é construído para definir as tarefas, sua distribuição aos indivíduos e equipes e quando tais tarefas deverão ser realizadas.

A Figura 7.2 mostra, resumidamente, as atividades de uma iteração do RUP, considerando apenas os fluxos centrados na engenharia.

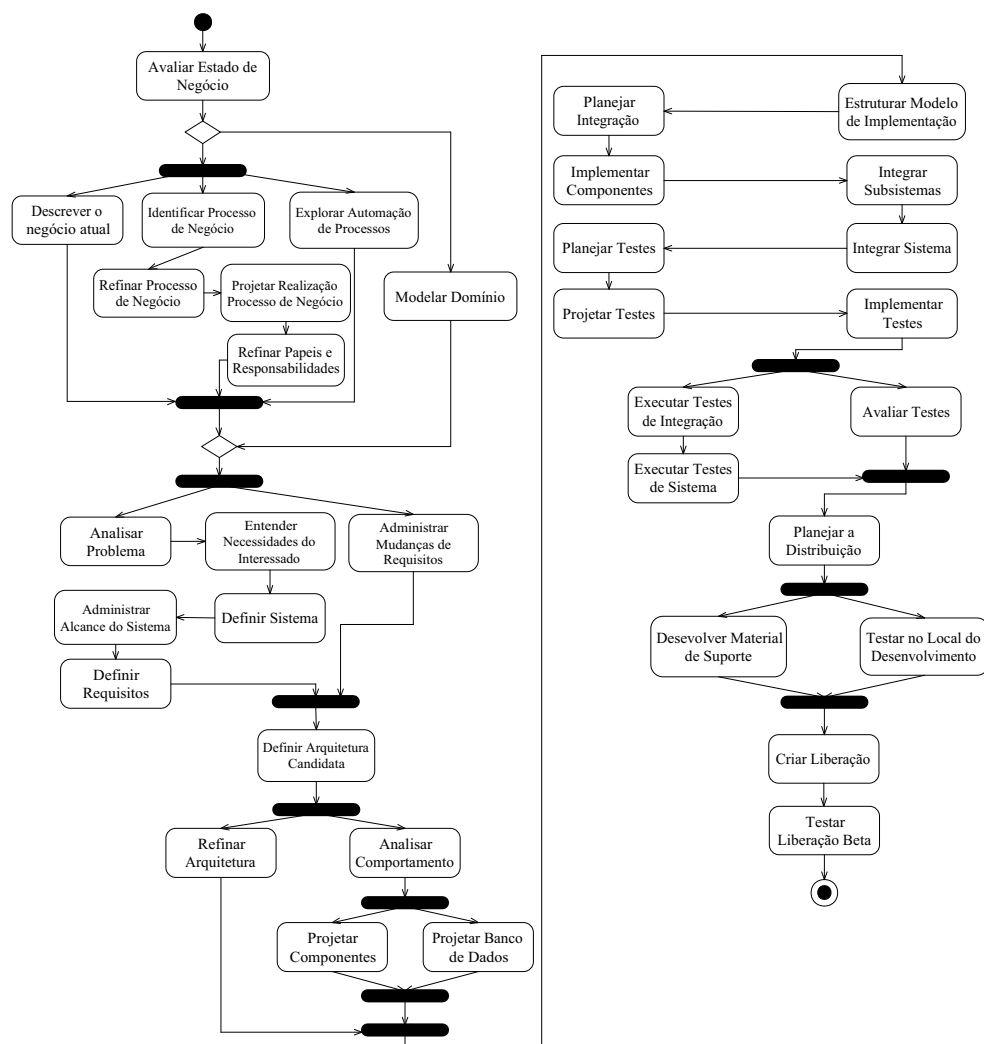


Figura 7.2: Diagrama de Atividades da UML que modela uma iteração do RUP.

7.1.2 Transformação do Diagrama de Atividades da UML que Modela uma Iteração do RUP em uma WorkFlow net

As regras de transformação mostradas na Seção 6.1.1 serão utilizadas para transformar o Diagrama de Atividades da UML mostrado na Figura 7.2 em uma WorkFlow net. A Figura 7.3 mostra a WorkFlow net correspondente, sendo que:

t1 = Avaliar Estado de Negócio,
t2 = Fork,
t3 = Descrever o Negócio Atual,
t4 = Identificar Processo de Negócio,
t5 = Explorar Automação de Processos,
t6 = Modelar Domínio,
t7 = Refinar Processo de Negócio,
t8 = Projetar Realização de Negócio,
t9 = Refinar Papéis e Responsabilidades,
t10 = Join,
t11 = Fork,
t12 = Administrar Mudanças de Requisitos,
t13 = Analisar Problema,
t14 = Entender Necessidades do Interessado,
t15 = Definir Sistema,
t16 = Administrar Alcance do Sistema,
t17 = Definir Requisitos,
t18 = Join,
t19 = Definir Arquitetura Candidata,
t20 = Fork,
t21 = Refinar Arquitetura,
t22 = Analisar Comportamento,
t23 = Fork,
t24 = Projetar Componentes,
t25 = Projetar Banco de Dados,
t26 = Join,
t27 = Join,
t28 = Estruturar Modelo de Implementação,
t29 = Planejar Integração,
t30 = Implementar Componentes,
t31 = Integrar Subsistemas,
t32 = Integrar Sistema,
t33 = Planejar Testes,

t34 = Projetar Testes,
 t35 = Implementar Testes,
 t36 = Fork,
 t37 = Executar Testes de Integração,
 t38 = Avaliar Testes,
 t39 = Executar Testes de Sistema,
 t40 = Join,
 t41 = Planejar a Distribuição,
 t42 = Fork,
 t43 = Desenvolver Material de Suporte,
 t44 = Testar o Produto no Local de Desenvolvimento,
 t45 = Join,
 t46 = Criar Liberação e
 t47 = Testar Liberação.

A partir desta transformação, pode-se analisar qualitativamente e quantitativamente esta WorkFlow net, utilizando as metodologias apresentadas nas Seções 5.1 e 5.2, respectivamente.

7.1.3 Análise Qualitativa: Verificação da Propriedade *Soundness* para uma Iteração do RUP

Como mostrado na Seção 5.1, para provar que uma WorkFlow net é *sound*, deve-se provar os seguintes da lógica linear que representam esta WorkFlow net e analisar as árvores de prova construídas para provar tais sequentes.

Para isso, inicialmente, é necessário transformar cada transição da WorkFlow net em uma fórmula da lógica linear. Esta transformação é mostrada a seguir:

t1 = Avaliar Estado de Negócio = $Start \multimap c1$,
 t2 = Fork = $c1 \multimap c2 \otimes c3 \otimes c4$,
 t3 = Descrever o Negócio Atual = $c3 \multimap c9$,
 t4 = Identificar Processo de Negócio = $c2 \multimap c5$,
 t5 = Explorar Automação de Processos = $c4 \multimap c10$,
 t6 = Modelar Domínio = $c1 \multimap c11$,
 t7 = Refinar Processo de Negócio = $c5 \multimap c6$,
 t8 = Projetar Realização de Negócio = $c6 \multimap c7$,
 t9 = Refinar Papéis e Responsabilidades = $c7 \multimap c8$,
 t10 = Join = $c8 \otimes c9 \otimes c10 \multimap c11$,
 t11 = Fork = $c11 \multimap c12 \otimes c13$,
 t12 = Administrar Mudanças de Requisitos = $c12 \multimap c19$,
 t13 = Analisar Problema = $c13 \multimap c14$,

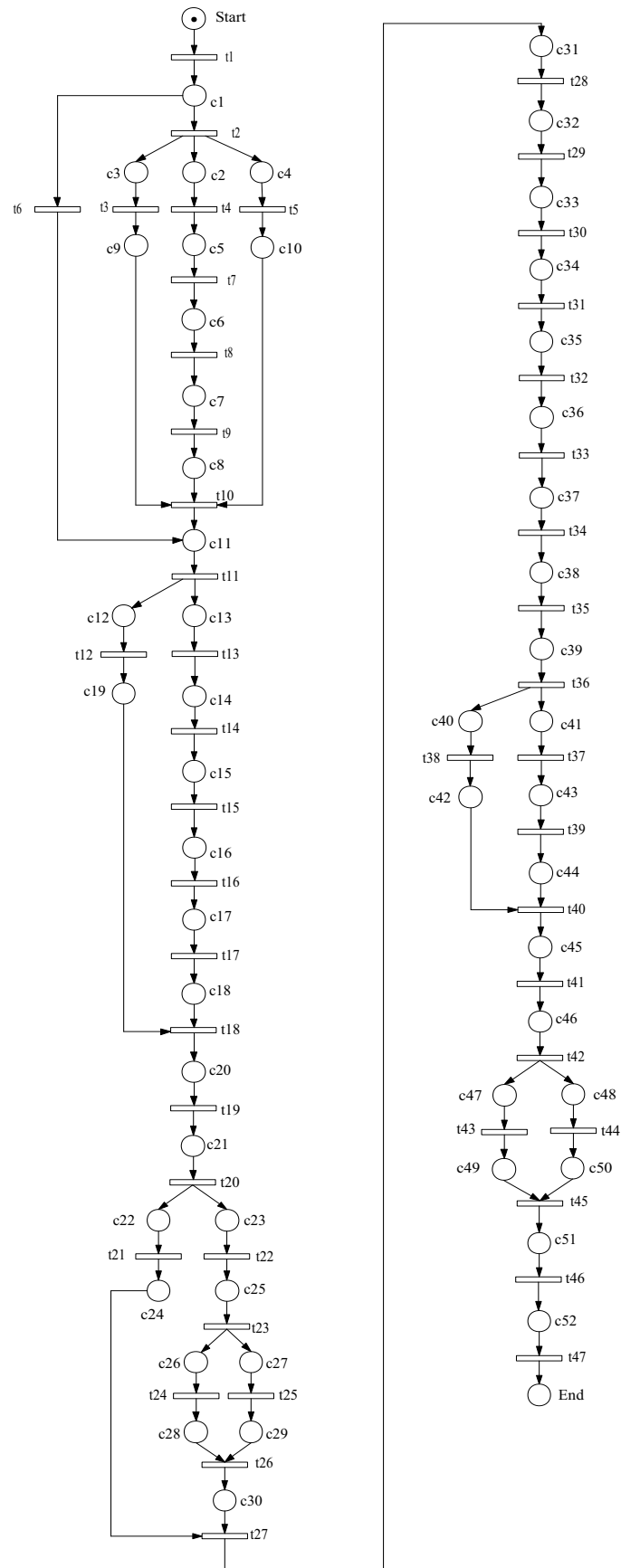


Figura 7.3: WorkFlow net que modela uma iteração do RUP.

$t14 = \text{Entender Necessidades do Interessado} = c14 \multimap c15,$
 $t15 = \text{Definir Sistema} = c15 \multimap c16,$
 $t16 = \text{Administrar Alcance do Sistema} = c16 \multimap c17,$
 $t17 = \text{Definir Requisitos} = c17 \multimap c18,$
 $t18 = \text{Join} = c18 \otimes c19 \multimap c20,$
 $t19 = \text{Definir Arquitetura Candidata} = c20 \multimap c21,$
 $t20 = \text{Fork} = c21 \multimap c22 \otimes c23,$
 $t21 = \text{Refinar Arquitetura} = c22 \multimap c24,$
 $t22 = \text{Analisar Comportamento} = c23 \multimap c25,$
 $t23 = \text{Fork} = c25 \multimap c26 \otimes c27,$
 $t24 = \text{Projetar Componentes} = c26 \multimap c28,$
 $t25 = \text{Projetar Banco de Dados} = c27 \multimap c29,$
 $t26 = \text{Join} = c28 \otimes c29 \multimap c30,$
 $t27 = \text{Join} = c24 \otimes c30 \multimap c31,$
 $t28 = \text{Estruturar Modelo de Implementação} = c31 \multimap c32,$
 $t29 = \text{Planejar Integração} = c32 \multimap c33,$
 $t30 = \text{Implementar Componentes} = c33 \multimap c34,$
 $t31 = \text{Integrar Subsistemas} = c34 \multimap c35,$
 $t32 = \text{Integrar Sistema} = c35 \multimap c36,$
 $t33 = \text{Planejar Testes} = c36 \multimap c37,$
 $t34 = \text{Projetar Testes} = c37 \multimap c38,$
 $t35 = \text{Implementar Testes} = c38 \multimap c39,$
 $t36 = \text{Fork} = c39 \multimap c40 \otimes c41,$
 $t37 = \text{Executar Testes de Integração} = c41 \multimap c43,$
 $t38 = \text{Avaliar Testes} = c40 \multimap c42,$
 $t39 = \text{Executar Testes de Sistema} = c43 \multimap c44,$
 $t40 = \text{Join} = c42 \otimes c44 \multimap c45,$
 $t41 = \text{Planejar a Distribuição} = c45 \multimap c46,$
 $t42 = \text{Fork} = c46 \multimap c47 \otimes c48,$
 $t43 = \text{Desenvolver Material de Suporte} = c47 \multimap c49,$
 $t44 = \text{Testar o Produto no Local de Desenvolvimento} = c48 \multimap c50,$
 $t45 = \text{Join} = c49 \otimes c50 \multimap c51,$
 $t46 = \text{Criar Liberação} = c51 \multimap c52,$
 $t47 = \text{Testar Liberação} = c52 \multimap \text{End}.$

Para provar a propriedade *soundness* para a WorkFlow net mostrada na Figura 7.3, dois sequentes deverão ser considerados, representando, respectivamente, os cenários Sc_1 e Sc_2 .

No cenário Sc_1 , as tarefas *Fork*, *Descrever o Negócio Atual*, *Identificar Processo de Negócio*, *Explorar Automação de Processos*, *Refinar Processo de Negócio*, *Projetar*

Realização de Negócio, Refinar Papéis e Responsabilidades e Join, representadas, respectivamente, pelas transições $t2, t3, t4, t5, t7, t8, t9$ e $t10$ da WorkFlow net mostrada na Figura 7.3 serão executadas. Este cenário é representado pelo seguinte abaixo:

$Start, t_1, t_2, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21},$
 $t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28}, t_{29}, t_{30}, t_{31}, t_{32}, t_{33}, t_{34}, t_{35}, t_{36}, t_{37}, t_{38}, t_{39}, t_{40}, t_{41}, t_{42},$
 $t_{43}, t_{44}, t_{45}, t_{46}, t_{47} \vdash End.$

Já no cenário Sc_2 , a tarefa *Modelar Domínio*, representada pela transição $t6$ da WorkFlow net mostrada na Figura 7.3, será executada. Este cenário é representado pelo seguinte abaixo:

$Start, t_1, t_6, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}, t_{27}, t_{28},$
 $t_{29}, t_{30}, t_{31}, t_{32}, t_{33}, t_{34}, t_{35}, t_{36}, t_{37}, t_{38}, t_{39}, t_{40}, t_{41}, t_{42}, t_{43}, t_{44}, t_{45}, t_{46}, t_{47} \vdash End.$

Para provar cada um dos sequentes definidos acima, uma árvore de prova canônica da lógica linear deverá ser construída. Na sequência, são apresentadas tais árvores de prova, onde $t_i \dots t_j$ representa as transições de i a j . Árvore de prova para o cenário Sc_1 :

$$\begin{array}{c}
\frac{c20 \vdash c20 \quad c21, c21 \multimap c22 \otimes c23, t_{21} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c18 \vdash c18 \quad c19 \vdash c19}{c18, c19 \vdash c18 \otimes c19} \otimes_R \quad c20, c20 \multimap c21, t_{20} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c17 \vdash c17 \quad c18, c19, c18 \otimes c19 \multimap c20, t_{19} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c16 \vdash c16 \quad c17, c19, c17 \multimap c18, t_{18} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c15 \vdash c15 \quad c16, c19, c16 \multimap c17, t_{17} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c14 \vdash c14 \quad c15, c19, c15 \multimap c16, t_{16} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c13 \vdash c13 \quad c14, c19, c14 \multimap c15, t_{15} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c12 \vdash c12 \quad c13, c19, c13 \multimap c14, t_{14} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c12, c13, c12 \multimap c19, t_{13} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c11 \vdash c11 \quad c12 \otimes c13, t_{12} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c8 \vdash c8 \quad c9 \vdash c9 \quad c10 \vdash c10}{c8, c9, c10 \vdash c8 \otimes c9 \otimes c10} \otimes_R \quad c11, c11 \multimap c12 \otimes c13, t_{12} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c7 \vdash c7 \quad c8, c9, c10, c8 \otimes c9 \otimes c10 \multimap c11, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c6 \vdash c6 \quad c7, c9, c10, c7 \multimap c8, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c5 \vdash c5 \quad c6, c9, c10, c6 \multimap c7, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c4 \vdash c4 \quad c5, c9, c10, c5 \multimap c6, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c2 \vdash c2 \quad c4, c5, c9, c4 \multimap c10, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c3 \vdash c3 \quad c2, c4, c9, c2 \multimap c5, t_5, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c2, c3, c4, c3 \multimap c9, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c1 \vdash c1 \quad c2 \otimes c3 \otimes c4, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{Start \vdash Start \quad c1, c1 \multimap c2 \otimes c3 \otimes c4, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
Start, Start \multimap c1, t_2, t_3, t_4, t_5, t_7, t_8, t_9, t_{10}, t_{11} \dots t_{47} \vdash End
\end{array}$$

Continuação da árvore de prova para o cenário Sc_1 (a partir do sequente $c21, c21 \multimap c22 \otimes c23, t_{21} \dots t_{47} \vdash \text{End}$):

$$\begin{array}{c}
\frac{c52 \vdash c52 \quad \text{End} \vdash \text{End}}{\multimap_L} \\
\frac{c51 \vdash c51 \quad c52, c52 \multimap \text{End} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c49 \vdash c49 \quad c50 \vdash c50}{c49, c50 \vdash c49 \otimes c50} \otimes_R \quad c51, c51 \multimap c52, t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c48 \vdash c48 \quad c49, c50, c49 \otimes c50 \multimap c51, t_{46}, t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c47 \vdash c47 \quad c48, c49, c48 \multimap c50, t_{45} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c47, c48, c47 \multimap c49, t_{44} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c46 \vdash c46 \quad c47 \otimes c48, t_{43} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c45 \vdash c45 \quad c46, c46 \multimap c47 \otimes c48, t_{43} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c42 \vdash c42 \quad c44 \vdash c44}{c42, c44 \vdash c42 \otimes c44} \otimes_R \quad c45, c45 \multimap c46, t_{42} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c43 \vdash c43 \quad c42, c44, c42 \otimes c44 \multimap c45, t_{41} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c40 \vdash c40 \quad c42, c43, c43 \multimap c44, t_{40} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c41 \vdash c41 \quad c40, c43, c40 \multimap c42, t_{39} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c40, c41, c41 \multimap c43, t_{38} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c39 \vdash c39 \quad c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c38 \vdash c38 \quad c39, c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c37 \vdash c37 \quad c38, c38 \multimap c39, t_{36} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c36 \vdash c36 \quad c37, c37 \multimap c38, t_{35} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c35 \vdash c35 \quad c36, c36 \multimap c37, t_{34} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c34 \vdash c34 \quad c35, c35 \multimap c36, t_{33} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c33 \vdash c33 \quad c34, c34 \multimap c35, t_{32} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c32 \vdash c32 \quad c33, c33 \multimap c34, t_{31} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c31 \vdash c31 \quad c32, c32 \multimap c33, t_{30} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c24 \vdash c24 \quad c30 \vdash c30}{c24, c30 \vdash c24 \otimes c30} \otimes_R \quad c31, c31 \multimap c32, t_{29} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c28 \vdash c28 \quad c29 \vdash c29}{c28, c29 \vdash c28 \otimes c29} \otimes_R \quad c24, c30, c24 \otimes c30 \multimap c31, t_{28} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c27 \vdash c27 \quad c24, c28, c29, c28 \otimes c29 \multimap c30, t_{27} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c26 \vdash c26 \quad c24, c27, c28, c27 \multimap c29, t_{26} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c24, c26, c27, c26 \multimap c28, t_{25} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c25 \vdash c25 \quad c24, c26 \otimes c27, t_{24} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c23 \vdash c23 \quad c24, c25, c25 \multimap c26 \otimes c27, t_{24} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c22 \vdash c22 \quad c23, c24, c23 \multimap c25, t_{23} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c22, c23, c22 \multimap c24, t_{22} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c21 \vdash c21 \quad c22 \otimes c23, t_{21} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
c21, c21 \multimap c22 \otimes c23, t_{21} \dots t_{47} \vdash \text{End}
\end{array}$$

Árvore de prova para o cenário Sc_2 :

$$\begin{array}{c}
\frac{c38 \vdash c38 \quad c39, c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c37 \vdash c37 \quad c38, c38 \multimap c39, t_{36} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c36 \vdash c36 \quad c37, c37 \multimap c38, t_{35} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c35 \vdash c35 \quad c36, c36 \multimap c37, t_{34} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c34 \vdash c34 \quad c35, c35 \multimap c36, t_{33} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c33 \vdash c33 \quad c34, c34 \multimap c35, t_{32} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c32 \vdash c32 \quad c33, c33 \multimap c34, t_{31} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c31 \vdash c31 \quad c32, c32 \multimap c33, t_{30} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c24 \vdash c24 \quad c30 \vdash c30}{c24, c30 \vdash c24 \otimes c30} \otimes_R \quad c31, c31 \multimap c32, t_{29} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c28 \vdash c28 \quad c29 \vdash c29}{c28, c29 \vdash c28 \otimes c29} \otimes_R \quad c24, c30, c24 \otimes c30 \multimap c31, t_{28} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c27 \vdash c27 \quad c24, c28, c29, c28 \otimes c29 \multimap c30, t_{27} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c26 \vdash c26 \quad c24, c27, c28, c27 \multimap c29, t_{26} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c24, c26, c27, c26 \multimap c28, t_{25} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c25 \vdash c25 \quad c24, c26 \otimes c27, t_{24} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c23 \vdash c23 \quad c24, c25, c25 \multimap c26 \otimes c27, t_{24} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c22 \vdash c22 \quad c23, c24, c23 \multimap c25, t_{23} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c22, c23, c22 \multimap c24, t_{22} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c21 \vdash c21 \quad c22 \otimes c23, t_{21} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c20 \vdash c20 \quad c21, c21 \multimap c22 \otimes c23, t_{21} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c18 \vdash c18 \quad c19 \vdash c19}{c18, c19 \vdash c18 \otimes c19} \otimes_R \quad c20, c20 \multimap c21, t_{20} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c17 \vdash c17 \quad c18, c19, c18 \otimes c19 \multimap c20, t_{19} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c16 \vdash c16 \quad c17, c19, c17 \multimap c18, t_{18} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c15 \vdash c15 \quad c16, c19, c16 \multimap c17, t_{17} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c14 \vdash c14 \quad c15, c19, c15 \multimap c16, t_{16} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c13 \vdash c13 \quad c14, c19, c14 \multimap c15, t_{15} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c12 \vdash c12 \quad c13, c19, c13 \multimap c14, t_{14} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c12, c13, c12 \multimap c19, t_{13} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c11 \vdash c11 \quad c12 \otimes c13, t_{12} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c1 \vdash c1 \quad c11, c11 \multimap c12 \otimes c13, t_{12} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\text{Start} \vdash \text{Start} \quad c1, c1 \multimap c11, t_{11} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\text{Start, Start} \multimap c1, t_6, t_{11} \dots t_{47} \vdash \text{End}
\end{array}$$

Continuação da árvore de prova para o cenário Sc_2 (prova a partir do sequente $c39, c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}$):

$$\begin{array}{c}
\frac{c52 \vdash c52 \quad \text{End} \vdash \text{End}}{\multimap_L} \\
\frac{c51 \vdash c51 \quad c52, c52 \multimap \text{End} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c49 \vdash c49 \quad c50 \vdash c50}{c49, c50 \vdash c49 \otimes c50} \otimes_R \quad c51, c51 \multimap c52, t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c48 \vdash c48 \quad c49, c50, c49 \otimes c50 \multimap c51, t_{46}, t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c47 \vdash c47 \quad c48, c49, c48 \multimap c50, t_{45} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c47, c48, c47 \multimap c49, t_{44} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c46 \vdash c46 \quad c47 \otimes c48, t_{43} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c45 \vdash c45 \quad c46, c46 \multimap c47 \otimes c48, t_{43} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{\frac{c42 \vdash c42 \quad c44 \vdash c44}{c42, c44 \vdash c42 \otimes c44} \otimes_R \quad c45, c45 \multimap c46, t_{42} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c43 \vdash c43 \quad c42, c44, c42 \otimes c44 \multimap c45, t_{41} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c40 \vdash c40 \quad c42, c43, c43 \multimap c44, t_{40} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c41 \vdash c41 \quad c40, c43, c40 \multimap c42, t_{39} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
\frac{c40, c41, c41 \multimap c43, t_{38} \dots t_{47} \vdash \text{End}}{\otimes_L} \\
\frac{c39 \vdash c39 \quad c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}}{\multimap_L} \\
c39, c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash \text{End}
\end{array}$$

Analisando as árvores de prova para os cenários Sc_1 e Sc_2 , conclui-se que a WorkFlow net da Figura 7.3 é *sound*, uma vez que:

- Apenas um átomo End foi produzido em cada cenário.
- Considerando as árvores de prova finalizadas, não há nenhum átomo disponível para consumo nestas árvores de prova.
- Considerando os cenários Sc_1 e Sc_2 da WorkFlow net analisada, cada transição desta WorkFlow net aparece em pelo menos um cenário e cada uma destas transições foi disparada. Assim, é possível afirmar que não há nenhuma transição morta na WorkFlow net da Figura 7.3.

7.1.4 Análise Quantitativa: Planejamento de Recursos para uma Iteração do RUP

Como descrito na Seção 5.2, o principal objetivo da análise quantitativa é calcular janelas de datas onde poderá ocorrer a utilização de um recurso para o tratamento de

atividades do processo de workflow analisado, podendo prever, assim, a disponibilidade dos recursos envolvidos na execução das tarefas correspondentes.

Para prover a análise quantitativa da WorkFlow net que modela uma iteração do RUP, é necessário incluir informações relativas ao tempo na WorkFlow net da Figura 7.3. Isso é feito associando um intervalo de tempo $[\delta_{i_{min}}, \delta_{i_{max}}]$ a cada transição $t_i \in T$, onde T é o conjunto de transições desta WorkFlow net.

A t-Time WorkFlow net que modela esse processo é mostrada na Figura 7.4.

Analisando a t-Time WorkFlow net da Figura 7.4, verifica-se que os cenários desta são os mesmos da WorkFlow net da Figura 7.3. Da mesma forma, observa-se que as transições da t-Time WorkFlow net da Figura 7.4 são as mesmas da WorkFlow net da Figura 7.3. Portanto, estas transições e cenários já foram definidos na Seção 7.1.3. Assim, árvores de prova canônica da lógica linear com cálculo de datas deverão ser construídas para cada cenário, Sc_1 e Sc_2 , possibilitando o cálculo de intervalos de datas simbólicas para a realização das tarefas presentes nesta t-Time WorkFlow net.

Na sequência é mostrada a árvore de prova canônica com cálculo de datas para o cenário Sc_1 . Para tal árvore de prova, considere que:

$$\begin{aligned}
 Seq_1 &= D_S + d_1 + d_2, \\
 Seq_2 &= d_4 + d_7 + d_8, \\
 Seq_3 &= Seq_1 + \max(d_3, d_5, Seq_2 + d_9) + d_{10}, \\
 Seq_4 &= d_{11} + d_{13} + d_{14} + d_{15}, \\
 Seq_5 &= Seq_3 + Seq_4, \\
 Seq_6 &= Seq_3 + \max(d_{11} + d_{12}, Seq_4 + d_{16} + d_{17}) + d_{18}, \\
 Seq_7 &= Seq_6 + d_{19} + d_{20} + d_{22} + d_{23}, \\
 Seq_8 &= Seq_7 + \max(d_{24}, d_{25}) + d_{26}, \\
 Seq_9 &= \max(Seq_6 + d_{19} + d_{20} + d_{21}, Seq_8) + d_{27}, \\
 Seq_{10} &= Seq_9 + d_{28} + d_{29} + d_{30}, \\
 Seq_{11} &= Seq_{10} + d_{31} + d_{32} + d_{33}, \\
 Seq_{12} &= Seq_{11} + d_{34} + d_{35} + d_{36}, \\
 Seq_{13} &= Seq_{12} + \max(d_{38}, d_{37} + d_{39}) + d_{40}, \\
 Seq_{14} &= Seq_{13} + d_{41} + d_{42}, \\
 Seq_{15} &= Seq_{14} + \max(d_{43}, d_{44}) + d_{45}.
 \end{aligned}$$

Árvore de prova com cálculo de datas para o cenário Sc_1 :

$$\begin{array}{c}
\frac{c24(Seq_6+d_{19}+d_{20}+d_{21},.),c26(Seq_7,.),c27(Seq_7,.),c26\multimap c28,t_{25}...t_{47}\vdash End}{\otimes_L} \\
\frac{c25(Seq_6+d_{19}+d_{20}+d_{22},Seq_7)\vdash c25 \quad c24(Seq_6+d_{19}+d_{20}+d_{21},.),c26(Seq_7,.)\otimes c27(Seq_7,.),t_{24}...t_{47}\vdash End}{\multimap_L} \\
\frac{c23(Seq_6+d_{19}+d_{20},Seq_7-d_{23})\vdash c23 \quad c24(Seq_6+d_{19}+d_{20}+d_{21},.),c25(Seq_6+d_{19}+d_{20}+d_{22},.),c25\multimap c26\otimes c27,t_{24}...t_{47}\vdash End}{\multimap_L} \\
\frac{c22(Seq_6+d_{19}+d_{20},Seq_6+d_{19}+d_{20}+d_{21})\vdash c22 \quad c23(Seq_6+d_{19}+d_{20},.),c24(Seq_6+d_{19}+d_{20}+d_{21},.),c23\multimap c25,t_{23}...t_{47}\vdash End}{\multimap_L} \\
\frac{c22(Seq_6+d_{19}+d_{20},.),c23(Seq_6+d_{19}+d_{20},.),c22\multimap c24,t_{22}...t_{47}\vdash End}{\otimes_L} \\
\frac{c21(Seq_6+d_{19},Seq_6+d_{19}+d_{20})\vdash c21 \quad c22(Seq_6+d_{19}+d_{20},.)\otimes c23(Seq_6+d_{19}+d_{20},.),t_{21}...t_{47}\vdash End}{\multimap_L} \\
\frac{c20(Seq_6,Seq_6+d_{19})\vdash c20 \quad c21(Seq_6+d_{19},.),c21\multimap c22\otimes c23,t_{21}...t_{47}\vdash End}{\multimap_L} \\
\frac{\frac{c18(Seq_5+d_{16}+d_{17},Seq_6)\vdash c18 \quad c19(Seq_3+d_{11}+d_{12},Seq_6)\vdash c19}{c18(Seq_5+d_{16}+d_{17},.),c19(Seq_3+d_{11}+d_{12},.)\vdash c18\otimes c19} \otimes_R \quad c20(Seq_6,.),c20\multimap c21,t_{20}...t_{47}\vdash End}{\multimap_L} \\
\frac{c17(Seq_5+d_{16},Seq_5+d_{16}+d_{17})\vdash c17 \quad c18(Seq_5+d_{16}+d_{17},.),c19(Seq_3+d_{11}+d_{12},.),c18\otimes c19\multimap c20,t_{19}...t_{47}\vdash End}{\multimap_L} \\
\frac{c16(Seq_5,Seq_5+d_{16})\vdash c16 \quad c17(Seq_5+d_{16},.),c19(Seq_3+d_{11}+d_{12},.),c17\multimap c18,t_{18}...t_{47}\vdash End}{\multimap_L} \\
\frac{c15(Seq_3+d_{11}+d_{13}+d_{14},Seq_5)\vdash c15 \quad c16(Seq_5,.),c19(Seq_3+d_{11}+d_{12},.),c16\multimap c17,t_{17}...t_{47}\vdash End}{\multimap_L} \\
\frac{c14(Seq_3+d_{11}+d_{13},Seq_3+d_{11}+d_{13}+d_{14})\vdash c14 \quad c15(Seq_3+d_{11}+d_{13}+d_{14},.),c19(Seq_3+d_{11}+d_{12},.),c15\multimap c16,t_{16}...t_{47}\vdash End}{\multimap_L} \\
\frac{c13(Seq_3+d_{11},Seq_3+d_{11}+d_{13})\vdash c13 \quad c14(Seq_3+d_{11}+d_{13},.),c19(Seq_3+d_{11}+d_{12},.),c14\multimap c15,t_{15}...t_{47}\vdash End}{\multimap_L} \\
\frac{c12(Seq_3+d_{11},Seq_3+d_{11}+d_{12})\vdash c12 \quad c13(Seq_3+d_{11},.),c19(Seq_3+d_{11}+d_{12},.),c13\multimap c14,t_{14}...t_{47}\vdash End}{\multimap_L} \\
\frac{c12(Seq_3+d_{11},.),c13(Seq_3+d_{11},.),c12\multimap c19,t_{13}...t_{47}\vdash End}{\otimes_L} \\
\frac{c11(Seq_3,Seq_3+d_{11})\vdash c11 \quad c12(Seq_3+d_{11},.)\otimes c13(Seq_3+d_{11},.),t_{12}...t_{47}\vdash End}{\multimap_L} \\
\frac{\frac{c8(Seq_1+Seq_2+d_9,Seq_3)\vdash c8 \quad c9(Seq_1+d_3,Seq_3)\vdash c9 \quad c10(Seq_1+d_5,Seq_3)\vdash c10}{c8(Seq_1+Seq_2+d_9,.),c9(Seq_1+d_3,.),c10(Seq_1+d_5,.)\vdash c8\otimes c9\otimes c10} \otimes_R \quad c11(Seq_3,.),c11\multimap c12\otimes c13,t_{12}...t_{47}\vdash End}{\multimap_L} \\
\frac{c7(Seq_1+Seq_2,Seq_1+Seq_2+d_9)\vdash c7 \quad c8(Seq_1+Seq_2+d_9,.),c9(Seq_1+d_3,.),c10(Seq_1+d_5,.),c8\otimes c9\otimes c10\multimap c11,t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c6(Seq_1+d_4+d_7,Seq_1+Seq_2)\vdash c6 \quad c7(Seq_1+Seq_2,.),c9(Seq_1+d_3,.),c10(Seq_1+d_5,.),c7\multimap c8,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c5(Seq_1+d_4,Seq_1+d_4+d_7)\vdash c5 \quad c6(Seq_1+d_4+d_7,.),c9(Seq_1+d_3,.),c10(Seq_1+d_5,.),c6\multimap c7,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c4(Seq_1,Seq_1+d_5)\vdash c4 \quad c5(Seq_1+d_4,.),c9(Seq_1+d_3,.),c10(Seq_1+d_5,.),c5\multimap c6,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c2(Seq_1,Seq_1+d_4)\vdash c2 \quad c4(Seq_1,.),c5(Seq_1+d_4,.),c9(Seq_1+d_3,.),c4\multimap c10,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c3(Seq_1,Seq_1+d_3)\vdash c3 \quad c2(Seq_1,.),c4(Seq_1,.),c9(Seq_1+d_3,.),c2\multimap c5,t_5,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{c2(Seq_1,.),c3(Seq_1,.),c4(Seq_1,.),c3\multimap c9,t_4,t_5,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\otimes_L} \\
\frac{c1(D_S+d_1,Seq_1)\vdash c1 \quad c2(Seq_1,.)\otimes c3(Seq_1,.)\otimes c4(Seq_1,.),t_3,t_4,t_5,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
\frac{Start(D_S,D_S+d_1)\vdash Start \quad c1(D_S+d_1,.),c1\multimap c2\otimes c3\otimes c4,t_3,t_4,t_5,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End}{\multimap_L} \\
Start(D_S,.),Start\multimap c1,t_2,t_3,t_4,t_5,t_7,t_8,t_9,t_{10},t_{11}...t_{47}\vdash End
\end{array}$$

Continuação da árvore de prova com cálculo de datas para o cenário Sc_1 (prova a partir do sequente $c24(Seq_6 + d_{19} + d_{20} + d_{21}, .), c26(Seq_7, .), c27(Seq_7, .), c26 \multimap c28, t_{25} \dots t_{47} \vdash End$):

$$\begin{array}{c}
\frac{c52(Seq_{15}+d_{46}, Seq_{15}+d_{46}+d_{47}) \vdash c52 \quad End(Seq_{15}+d_{46}+d_{47}, .) \vdash End}{\multimap_L} \\
\frac{c51(Seq_{15}, Seq_{15}+d_{46}) \vdash c51 \quad c52(Seq_{15}+d_{46}, .), c52 \multimap End \vdash End}{\multimap_L} \\
\frac{\frac{c49(Seq_{14}+d_{43}, Seq_{15}) \vdash c49 \quad c50(Seq_{14}+d_{44}, Seq_{15}) \vdash c50}{c49(Seq_{14}+d_{43}, .), c50(Seq_{14}+d_{44}, .) \vdash c49 \otimes c50} \otimes_R \quad c51(Seq_{15}, .), c51 \multimap c52, t_{47} \vdash End}{\multimap_L} \\
\frac{c48(Seq_{14}, Seq_{14}+d_{44}) \vdash c48 \quad c49(Seq_{14}+d_{43}, .), c50(Seq_{14}+d_{44}, .), c49 \otimes c50 \multimap c51, t_{46}, t_{47} \vdash End}{\multimap_L} \\
\frac{c47(Seq_{14}, Seq_{14}+d_{43}) \vdash c47 \quad c48(Seq_{14}, .), c49(Seq_{14}+d_{43}, .), c48 \multimap c50, t_{45} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c47(Seq_{14}, Seq_{14}+d_{43}), c48(Seq_{14}, .), c47 \multimap c49, t_{44} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c46(Seq_{13}+d_{41}, Seq_{14}) \vdash c46 \quad c47(Seq_{14}, .) \otimes c48(Seq_{14}, .), t_{43} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c45(Seq_{13}, Seq_{13}+d_{41}) \vdash c45 \quad c46(Seq_{13}+d_{41}, .), c46 \multimap c47 \otimes c48, t_{43} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c42(Seq_{12}+d_{38}, Seq_{13}) \vdash c42 \quad c44(Seq_{12}+d_{37}+d_{39}, Seq_{13}) \vdash c44}{c42(Seq_{12}+d_{38}, .), c44(Seq_{12}+d_{37}+d_{39}, .) \vdash c42 \otimes c44} \otimes_R \quad c45(Seq_{13}, .), c45 \multimap c46, t_{42} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c43(Seq_{12}+d_{37}, Seq_{12}+d_{37}+d_{39}) \vdash c43 \quad c42(Seq_{12}+d_{38}, .), c44(Seq_{12}+d_{37}+d_{39}, .), c42 \otimes c44 \multimap c45, t_{41} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c40(Seq_{12}, Seq_{12}+d_{38}) \vdash c40 \quad c42(Seq_{12}+d_{38}, .), c43(Seq_{12}+d_{37}, .), c43 \multimap c44, t_{40} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c41(Seq_{12}, Seq_{12}+d_{37}) \vdash c41 \quad c40(Seq_{12}, .), c43(Seq_{12}+d_{37}, .), c40 \multimap c42, t_{39} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c40(Seq_{12}, .), c41(Seq_{12}, .), c41 \multimap c43, t_{38} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c39(Seq_{11}+d_{34}+d_{35}, Seq_{12}) \vdash c39 \quad c40(Seq_{12}, .) \otimes c41(Seq_{12}, .), t_{37} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c38(Seq_{11}+d_{34}, Seq_{11}+d_{34}+d_{35}) \vdash c38 \quad c39(Seq_{11}+d_{34}+d_{35}, .), c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c37(Seq_{11}, Seq_{11}+d_{34}) \vdash c37 \quad c38(Seq_{11}+d_{34}, .), c38 \multimap c39, t_{36} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c36(Seq_{10}+d_{31}+d_{32}, Seq_{11}) \vdash c36 \quad c37(Seq_{11}, .), c37 \multimap c38, t_{35} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c35(Seq_{10}+d_{31}, Seq_{10}+d_{31}+d_{32}) \vdash c35 \quad c36(Seq_{10}+d_{31}+d_{32}, .), c36 \multimap c37, t_{34} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c34(Seq_{10}, Seq_{10}+d_{31}) \vdash c34 \quad c35(Seq_{10}+d_{31}, .), c35 \multimap c36, t_{33} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c33(Seq_9+d_{28}+d_{29}, Seq_{10}) \vdash c33 \quad c34(Seq_{10}, .), c34 \multimap c35, t_{32} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c32(Seq_9+d_{28}, Seq_9+d_{28}+d_{29}) \vdash c32 \quad c33(Seq_9+d_{28}+d_{29}, .), c33 \multimap c34, t_{31} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c31(Seq_9, Seq_9+d_{28}) \vdash c31 \quad c32(Seq_9+d_{28}, .), c32 \multimap c33, t_{30} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c24(Seq_6+d_{19}+d_{20}+d_{21}, Seq_9) \vdash c24 \quad c30(Seq_8, Seq_9) \vdash c30}{c24(Seq_6+d_{19}+d_{20}+d_{21}, .), c30(Seq_8, .) \vdash c24 \otimes c30} \otimes_R \quad c31(Seq_9, .), c31 \multimap c32, t_{29} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c28(Seq_7+d_{24}, Seq_8) \vdash c28 \quad c29(Seq_7+d_{25}, Seq_8) \vdash c29}{c28(Seq_7+d_{24}, .), c29(Seq_7+d_{25}, .) \vdash c28 \otimes c29} \otimes_R \quad c24(Seq_6+d_{19}+d_{20}+d_{21}, .), c30(Seq_8, .), c24 \otimes c30 \multimap c31, t_{28} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c27(Seq_7, Seq_7+d_{25}) \vdash c27 \quad c24(Seq_6+d_{19}+d_{20}+d_{21}, .), c28(Seq_7+d_{24}, .), c29(Seq_7+d_{25}, .), c28 \otimes c29 \multimap c30, t_{27} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c26(Seq_7, Seq_7+d_{24}) \vdash c26 \quad c24(Seq_6+d_{19}+d_{20}+d_{21}, .), c27(Seq_7, .), c28(Seq_7+d_{24}, .), c27 \multimap c29, t_{26} \dots t_{47} \vdash End}{\multimap_L} \\
c24(Seq_6+d_{19}+d_{20}+d_{21}, .), c26(Seq_7, .), c27(Seq_7, .), c26 \multimap c28, t_{25} \dots t_{47} \vdash End
\end{array}$$

Na sequência, é apresentada a árvore de prova com cálculo de datas para o cenário Sc_2 da t-Time WorkFlow net da Figura 7.4. Para tanto, considere que:

$$\begin{aligned}
Seq_1 &= D_S + d_1 + d_6, \\
Seq_2 &= d_{11} + d_{13} + d_{14} + d_{15}, \\
Seq_3 &= Seq_1 + Seq_2, \\
Seq_4 &= Seq_1 + \max(d_{11} + d_{12}, Seq_2 + d_{16} + d_{17}) + d_{18}, \\
Seq_5 &= Seq_4 + d_{19} + d_{20} + d_{22} + d_{23}, \\
Seq_6 &= Seq_5 + \max(d_{24}, d_{25}) + d_{26}, \\
Seq_7 &= \max(Seq_4 + d_{19} + d_{20} + d_{21}, Seq_6) + d_{27}, \\
Seq_8 &= Seq_7 + d_{28} + d_{29} + d_{30}, \\
Seq_9 &= Seq_8 + d_{31} + d_{32} + d_{33}, \\
Seq_{10} &= Seq_9 + d_{34} + d_{35} + d_{36}, \\
Seq_{11} &= Seq_{10} + \max(d_{38}, d_{37} + d_{39}) + d_{40}, \\
Seq_{12} &= Seq_{11} + d_{41} + d_{42}, \\
Seq_{13} &= Seq_{12} + \max(d_{43}, d_{44}) + d_{45}.
\end{aligned}$$

Árvore de prova com cálculo de datas para o cenário Sc_2 :

$$\begin{aligned}
&\frac{c22(Seq_4 + d_{19} + d_{20}, Seq_4 + d_{19} + d_{20} + d_{21}) \vdash c22 \quad c23(Seq_4 + d_{19} + d_{20}, .), c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c23 \multimap c25, t_{23} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c22(Seq_4 + d_{19} + d_{20}, .), c23(Seq_4 + d_{19} + d_{20}, .), c22 \multimap c24, t_{22} \dots t_{47} \vdash End}{\otimes_L} \\
&\frac{c21(Seq_4 + d_{19}, Seq_4 + d_{19} + d_{20}) \vdash c21 \quad c22(Seq_4 + d_{19} + d_{20}, .) \otimes c23(Seq_4 + d_{19} + d_{20}, .), t_{21} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c20(Seq_4, Seq_4 + d_{19}) \vdash c20 \quad c21(Seq_4 + d_{19}, .), c21 \multimap c22 \otimes c23, t_{21} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{\frac{c18(Seq_3 + d_{16} + d_{17}, Seq_4) \vdash c18 \quad c19(Seq_1 + d_{11} + d_{12}, Seq_4) \vdash c19}{c18(Seq_3 + d_{16} + d_{17}, .), c19(Seq_1 + d_{11} + d_{12}, .) \vdash c18 \otimes c19} \otimes_R \quad c20(Seq_4, .), c20 \multimap c21, t_{20} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c17(Seq_3 + d_{16}, Seq_3 + d_{16} + d_{17}) \vdash c17 \quad c18(Seq_3 + d_{16} + d_{17}, .), c19(Seq_1 + d_{11} + d_{12}, .), c18 \otimes c19 \multimap c20, t_{19} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c16(Seq_3, Seq_3 + d_{16}) \vdash c16 \quad c17(Seq_3 + d_{16}, .), c19(Seq_1 + d_{11} + d_{12}, .), c17 \multimap c18, t_{18} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c15(Seq_1 + d_{11} + d_{13} + d_{14}, Seq_3) \vdash c15 \quad c16(Seq_3, .), c19(Seq_1 + d_{11} + d_{12}, .), c16 \multimap c17, t_{17} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c14(Seq_1 + d_{11} + d_{13}, Seq_1 + d_{11} + d_{13} + d_{14}) \vdash c14 \quad c15(Seq_1 + d_{11} + d_{13} + d_{14}, .), c19(Seq_1 + d_{11} + d_{12}, .), c15 \multimap c16, t_{16} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c13(Seq_1 + d_{11}, Seq_1 + d_{11} + d_{13}) \vdash c13 \quad c14(Seq_1 + d_{11} + d_{13}, .), c19(Seq_1 + d_{11} + d_{12}, .), c14 \multimap c15, t_{15} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c12(Seq_1 + d_{11}, Seq_1 + d_{11} + d_{12}) \vdash c12 \quad c13(Seq_1 + d_{11}, .), c19(Seq_1 + d_{11} + d_{12}, .), c13 \multimap c14, t_{14} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c12(Seq_1 + d_{11}, .), c13(Seq_1 + d_{11}, .), c12 \multimap c19, t_{13} \dots t_{47} \vdash End}{\otimes_L} \\
&\frac{c11(Seq_1, Seq_1 + d_{11}) \vdash c11 \quad c12(Seq_1 + d_{11}, .) \otimes c13(Seq_1 + d_{11}, .), t_{12} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{c1(D_S + d_1, Seq_1) \vdash c1 \quad c11(Seq_1, .), c11 \multimap c12 \otimes c13, t_{12} \dots t_{47} \vdash End}{\multimap_L} \\
&\frac{Start(D_S, D_S + d_1) \vdash Start \quad c1(D_S + d_1, .), c1 \multimap c11, t_{11} \dots t_{47} \vdash End}{\multimap_L} \\
&Start(D_S, .), Start \multimap c1, t_6, t_{11} \dots t_{47} \vdash End
\end{aligned}$$

Continuação da árvore de prova com cálculo de datas para o cenário Sc_2 (prova a partir do sequente $c23(Seq_4 + d_{19} + d_{20}, .), c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c23 \multimap c25, t_{23} \dots t_{47} \vdash End$):

$$\begin{array}{c}
\frac{c52(Seq_{13} + d_{46}, Seq_{13} + d_{46} + d_{47}) \vdash c52 \quad End(Seq_{13} + d_{46} + d_{47}, .) \vdash End}{\multimap_L} \\
\frac{c51(Seq_{13}, Seq_{13} + d_{46}) \vdash c51 \quad c52(Seq_{13} + d_{46}, .), c52 \multimap End \vdash End}{\multimap_L} \\
\frac{\frac{c49(Seq_{12} + d_{43}, Seq_{13}) \vdash c49 \quad c50(Seq_{12} + d_{44}, Seq_{13}) \vdash c50}{c49(Seq_{12} + d_{43}, .), c50(Seq_{12} + d_{44}, .) \vdash c49 \otimes c50} \otimes_R \quad c51(Seq_{13}, .), c51 \multimap c52, t_{47} \vdash End}{\multimap_L} \\
\frac{c48(Seq_{12}, Seq_{12} + d_{44}) \vdash c48 \quad c49(Seq_{12} + d_{43}, .), c50(Seq_{12} + d_{44}, .), c49 \otimes c50 \multimap c51, t_{46}, t_{47} \vdash End}{\multimap_L} \\
\frac{c47(Seq_{12}, Seq_{12} + d_{43}) \vdash c47 \quad c48(Seq_{12}, .), c49(Seq_{12} + d_{43}, .), c48 \multimap c50, t_{45} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c47(Seq_{12}, Seq_{12} + d_{43}), c48(Seq_{12}, .), c47 \multimap c49, t_{44} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c46(Seq_{11} + d_{41}, Seq_{12}) \vdash c46 \quad c47(Seq_{12}, .) \otimes c48(Seq_{12}, .), t_{43} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c45(Seq_{11}, Seq_{11} + d_{41}) \vdash c45 \quad c46(Seq_{11} + d_{41}, .), c46 \multimap c47 \otimes c48, t_{43} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c42(Seq_{10} + d_{38}, Seq_{11}) \vdash c42 \quad c44(Seq_{10} + d_{37} + d_{39}, Seq_{11}) \vdash c44}{c42(Seq_{10} + d_{38}, .), c44(Seq_{10} + d_{37} + d_{39}, .) \vdash c42 \otimes c44} \otimes_R \quad c45(Seq_{11}, .), c45 \multimap c46, t_{42} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c43(Seq_{10} + d_{37}, Seq_{10} + d_{37} + d_{39}) \vdash c43 \quad c42(Seq_{10} + d_{38}, .), c44(Seq_{10} + d_{37} + d_{39}, .), c42 \otimes c44 \multimap c45, t_{41} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c40(Seq_{10}, Seq_{10} + d_{38}) \vdash c40 \quad c42(Seq_{10} + d_{38}, .), c43(Seq_{10} + d_{37}, .), c43 \multimap c44, t_{40} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c41(Seq_{10}, Seq_{10} + d_{37}) \vdash c41 \quad c40(Seq_{10}, .), c43(Seq_{10} + d_{37}, .), c40 \multimap c42, t_{39} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c40(Seq_{10}, .), c41(Seq_{10}, .), c41 \multimap c43, t_{38} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c39(Seq_9 + d_{34} + d_{35}, Seq_{10}) \vdash c39 \quad c40(Seq_{10}, .) \otimes c41(Seq_{10}, .), t_{37} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c38(Seq_9 + d_{34}, Seq_9 + d_{34} + d_{35}) \vdash c38 \quad c39(Seq_9 + d_{34} + d_{35}, .), c39 \multimap c40 \otimes c41, t_{37} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c37(Seq_9, Seq_9 + d_{34}) \vdash c37 \quad c38(Seq_9 + d_{34}, .), c38 \multimap c39, t_{36} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c36(Seq_8 + d_{31} + d_{32}, Seq_9) \vdash c36 \quad c37(Seq_9, .), c37 \multimap c38, t_{35} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c35(Seq_8 + d_{31}, Seq_8 + d_{31} + d_{32}) \vdash c35 \quad c36(Seq_8 + d_{31} + d_{32}, .), c36 \multimap c37, t_{34} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c34(Seq_8, Seq_8 + d_{31}) \vdash c34 \quad c35(Seq_8 + d_{31}, .), c35 \multimap c36, t_{33} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c33(Seq_7 + d_{28} + d_{29}, Seq_8) \vdash c33 \quad c34(Seq_8, .), c34 \multimap c35, t_{32} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c32(Seq_7 + d_{28}, Seq_7 + d_{28} + d_{29}) \vdash c32 \quad c33(Seq_7 + d_{28} + d_{29}, .), c33 \multimap c34, t_{31} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c31(Seq_7, Seq_7 + d_{28}) \vdash c31 \quad c32(Seq_7 + d_{28}, .), c32 \multimap c33, t_{30} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c24(Seq_4 + d_{19} + d_{20} + d_{21}, Seq_7) \vdash c24 \quad c30(Seq_6, Seq_7) \vdash c30}{c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c30(Seq_6, .) \vdash c24 \otimes c30} \otimes_R \quad c31(Seq_7, .), c31 \multimap c32, t_{29} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{\frac{c28(Seq_5 + d_{24}, Seq_6) \vdash c28 \quad c29(Seq_5 + d_{25}, Seq_6) \vdash c29}{c28(Seq_5 + d_{24}, .), c29(Seq_5 + d_{25}, .) \vdash c28 \otimes c29} \otimes_R \quad c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c30(Seq_6, .), c24 \otimes c30 \multimap c31, t_{28} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c27(Seq_5, Seq_5 + d_{25}) \vdash c27 \quad c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c28(Seq_5 + d_{24}, .), c29(Seq_5 + d_{25}, .), c28 \otimes c29 \multimap c30, t_{27} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c26(Seq_5, Seq_5 + d_{24}) \vdash c26 \quad c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c27(Seq_5, .), c28(Seq_5 + d_{24}, .), c27 \multimap c29, t_{26} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c26(Seq_5, .), c27(Seq_5, .), c26 \multimap c28, t_{25} \dots t_{47} \vdash End}{\otimes_L} \\
\frac{c25(Seq_4 + d_{19} + d_{20} + d_{22}, Seq_5) \vdash c25 \quad c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c26(Seq_5, .) \otimes c27(Seq_5, .), t_{24} \dots t_{47} \vdash End}{\multimap_L} \\
\frac{c23(Seq_4 + d_{19} + d_{20}, Seq_5 - d_{23}) \vdash c23 \quad c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c25(Seq_4 + d_{19} + d_{20} + d_{22}, .), c25 \multimap c26 \otimes c27, t_{24} \dots t_{47} \vdash End}{\multimap_L} \\
c23(Seq_4 + d_{19} + d_{20}, .), c24(Seq_4 + d_{19} + d_{20} + d_{21}, .), c23 \multimap c25, t_{23} \dots t_{47} \vdash End
\end{array}$$

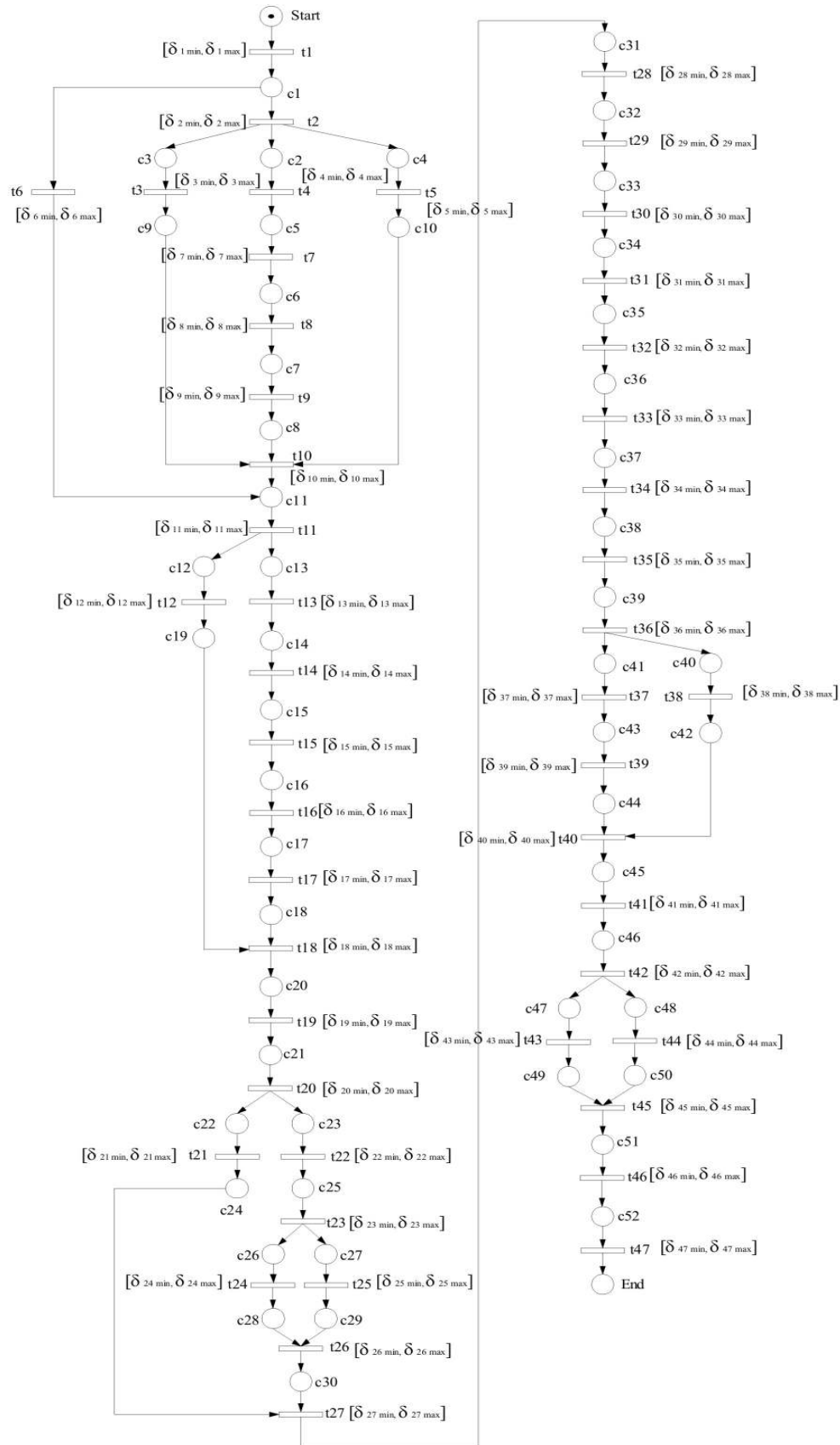


Figura 7.4: t-Time Workflow net, com datas simbólicas, que modela uma iteração do RUP.

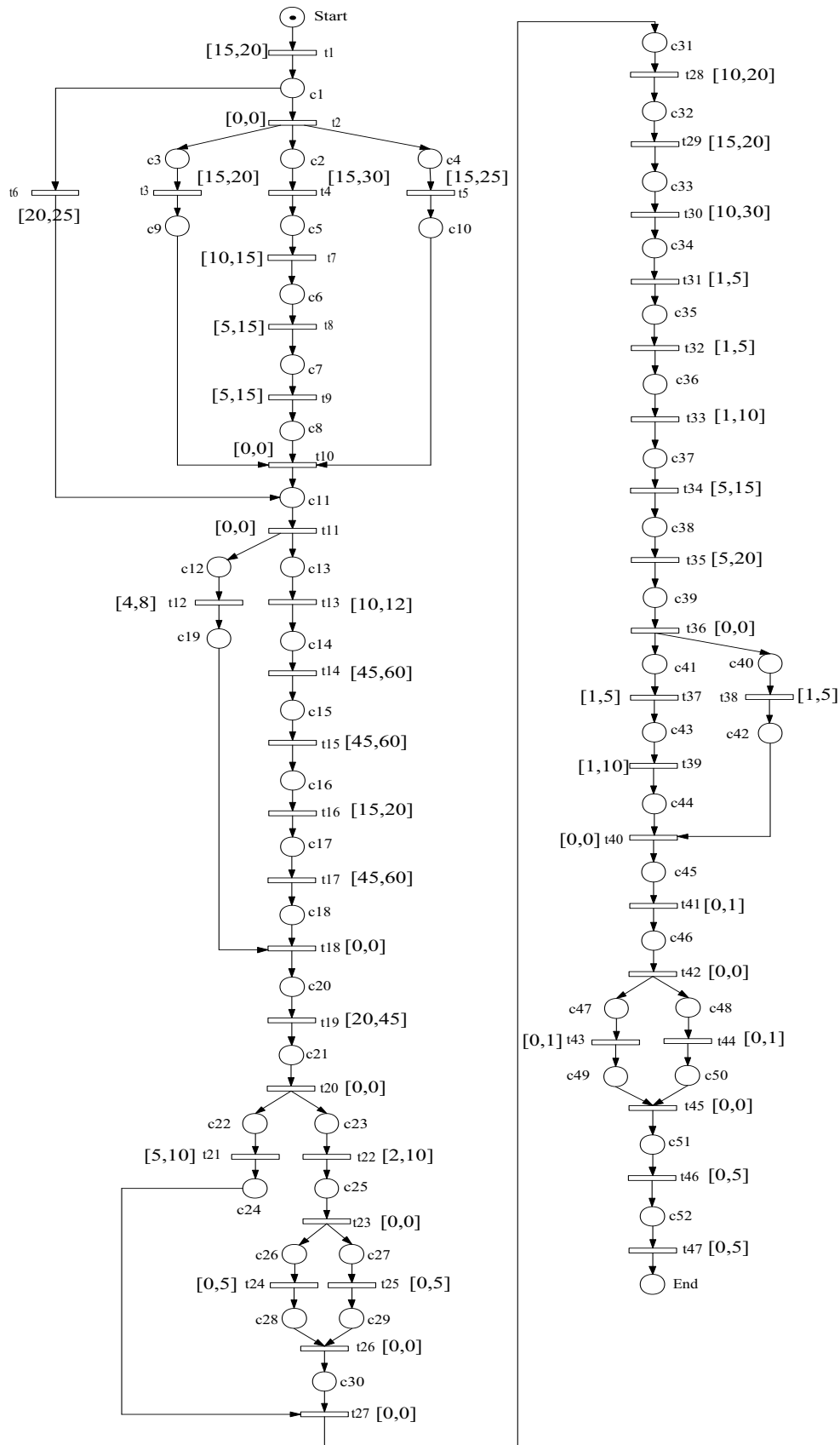


Figura 7.5: t-Time WorkFlow net, com datas numéricas, que modela uma iteração do RUP.

Uma vez que as árvores de prova canônica com cálculo de datas estão construídas, basta extrair as datas de produção, D_P , e consumo, D_C , do átomo que representa a pré-condição da transição correspondente a tarefa para a qual se deseja calcular o intervalo de execução. Quando há mais de uma pré-condição associada à transição, considera-se a data máxima das produções dos átomos correspondentes a estas pré-condições. Como já apresentado na Seção 5.2, a data de produção do átomo, D_P , corresponde ao início da execução da tarefa associada à transição e a data de consumo, D_C , corresponde ao término da execução da mesma. Assim, é gerado um intervalo $[D_P, D_C]$ de datas onde o recurso que executará a referida tarefa poderá ser requisitado para realizá-la.

Vale ressaltar que, uma vez que as datas de produção e consumo são dependentes de durações de sensibilização d_i , cujo valor pertence a um intervalo de tempo $\Delta_i = [\delta_{imin}, \delta_{imax}]$, pode-se considerar vários intervalos possíveis de execução das tarefas, de acordo com um planejamento estratégico. Por exemplo, o intervalo de execução $I_{Exec} = [D_{Pmin}, D_{Cmax}]$ considera que a alocação do recurso para a execução da tarefa poderá ocorrer entre o início ao mais cedo e o término ao mais tarde da atividade considerada.

A informação sobre quando o processo global finalizará, para um dado caso, é dado pela data de produção do átomo End . Assim, pode-se calcular a data ao mais cedo de finalização do processo, dada pela data mínima D_{Pmin} de produção deste átomo e a data ao mais tarde de finalização do processo, dada pela data máxima D_{Pmax} de produção do átomo End .

Considerando a t-Time WorkFlow net da Figura 7.4, a data de término de uma iteração do RUP, considerando o cenário Sc_2 , é dada por: $Seq_{13} + d_{46} + d_{47}$. Realizando as substituições necessárias, tem-se que a data de término é dada por: $D_S + d_1 + d_6 + d_{11} + \max(d_{12}, d_{13} + d_{14} + d_{15} + d_{16} + d_{17}) + d_{18} + d_{19} + d_{20} + \max(d_{21}, d_{22} + d_{23} + \max(d_{24}, d_{25}) + d_{26}) + d_{27} + d_{28} + d_{29} + d_{30} + d_{31} + d_{32} + d_{33} + d_{34} + d_{35} + d_{36} + \max(d_{38}, d_{37} + d_{39}) + d_{40} + d_{41} + d_{42} + \max(d_{43}, d_{44}) + d_{45} + d_{46} + d_{47}$.

Uma vez que d_i tem valor pertencente a um intervalo de tempo $\Delta_i = [\delta_{imin}, \delta_{imax}]$ e considerando que deseja-se saber as datas de término ao mais cedo e ao mais tarde de uma dada iteração do RUP, tem-se:

- Data de término ao mais cedo de uma iteração do RUP: $D_S + d_{1min} + d_{6min} + d_{11min} + \max(d_{12min}, d_{13min} + d_{14min} + d_{15min} + d_{16min} + d_{17min}) + d_{18min} + d_{19min} + d_{20min} + \max(d_{21min}, d_{22min} + d_{23min} + \max(d_{24min}, d_{25min}) + d_{26min}) + d_{27min} + d_{28min} + d_{29min} + d_{30min} + d_{31min} + d_{32min} + d_{33min} + d_{34min} + d_{35min} + d_{36min} + \max(d_{38min}, d_{37min} + d_{39min}) + d_{40min} + d_{41min} + d_{42min} + \max(d_{43min}, d_{44min}) + d_{45min} + d_{46min} + d_{47min}$.
- Data de término ao mais tarde de uma iteração do RUP: $D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max} + d_{19max} + d_{20max} + \max(d_{21max}, d_{22max} + d_{23max} + \max(d_{24max}, d_{25max}) + d_{26max}) + d_{27max} + d_{28max} + d_{29max} + d_{30max} + d_{31max} + d_{32max} + d_{33max} + d_{34max} + d_{35max} + d_{36max} + \max(d_{38max}, d_{37max} +$

$$d_{39_{max}}) + d_{40_{max}} + d_{41_{max}} + d_{42_{max}} + \max(d_{43_{max}}, d_{44_{max}}) + d_{45_{max}} + d_{46_{max}} + d_{47_{max}}.$$

Uma vez que as datas calculadas são datas simbólicas, estas podem ser utilizadas para todas as iterações desejadas, alterando-se apenas os valores das datas. Por exemplo, considerando hipoteticamente que uma dada iteração do RUP inicie-se na data 0, ou seja, $D_S = 0$, que cada tarefa dessa iteração tenha as durações mostradas na t-Time WorkFlow net da Figura 7.5 e que estas durações representam dias, então, tem-se que a data máxima de finalização é dada pela fórmula acima, substituindo-se as datas simbólicas pelas datas numéricas mostradas nesta t-Time WorkFlow net, ou seja:

$$\begin{aligned} D_{P_{maxEnd}} &= 0 + 20 + 25 + 0 + \max(8, 12 + 60 + 60 + 20 + 60) + 0 + 45 + 0 + \max(10, 10 + 0 + \\ &\max(5, 5) + 0) + 0 + 20 + 20 + 30 + 5 + 5 + 10 + 15 + 20 + 0 + \max(5, 5 + 10) + 0 + 1 + 0 + \max(1, 1) + \\ &0 + 5 + 5 = 45 + \max(8, 212) + 45 + \max(10, 15) + 125 + \max(5, 15) + \max(1, 1) + 10 = \\ &45 + 212 + 45 + 15 + 125 + 15 + 1 + 10 = 468. \end{aligned}$$

Da mesma forma que foram extraídas as datas mínima e máxima de finalização da iteração do RUP, podem ser extraídas as datas de execução de qualquer atividade presente na t-Time Workflow net da Figura 7.4. Por exemplo, considere que deseja-se saber quando a tarefa *Entender Necessidades do Interessado*, representada pela transição t_{14} da t-Time WorkFlow net da Figura 7.4, será executada, sendo que a atividade poderá se iniciar ao mais cedo e se finalizar ao mais tarde, ou seja, $I_{Exec} = [D_{P_{min}}, D_{C_{max}}]$. Supondo que a iteração do RUP analisada inicie-se na data 0, ou seja, $D_S = 0$ e que esta iteração considera o cenário Sc_2 , onde a tarefa *Modelar Domínio* será executada, então o intervalo de execução da tarefa *Entender Necessidades do Interessado* é dado por: $I_{Exec} = [D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}} + d_{13_{min}}, D_S + d_{1_{max}} + d_{6_{max}} + d_{11_{max}} + d_{13_{max}} + d_{14_{max}}]$. Realizando as devidas substituições, de acordo com as datas numéricas mostradas na t-Time WorkFlow net da Figura 7.5, tem-se que $I_{Exec} = [0 + 15 + 20 + 0 + 10, 0 + 20 + 25 + 0 + 12 + 60] = [45, 117]$. Ou seja, a atividade *Entender Necessidades do Interessado* será se iniciará ao mais cedo na data 45 e se finalizará ao mais tarde na data 117.

Este mesmo processo pode ser realizado para encontrar o intervalo de execução de cada tarefa mapeada na t-Time WorkFlow net analisada.

Como citado anteriormente, as atividades presentes no Diagrama de Atividades da UML da Figura 7.2 representam apenas as atividades do RUP que são centradas no fluxo de engenharia. O início da atividade *Avaliar Estado de Negócio* marca o início do fluxo de modelagem de negócio. O início das atividades *Analisar Problema* e *Administrar Mudança de Requisitos* marcam o término do fluxo de modelagem de negócio e o início do fluxo de requisitos. Já o início da atividade *Definir Arquitetura Candidata* marca o término do fluxo de requisitos e o início do fluxo de análise e design. Desta mesma forma, o início da atividade *Estruturar Modelo de Implementação* marca o início do fluxo de implementação e o término do fluxo de análise e projeto. Já o início da atividade *Planejar Testes* marca o término do fluxo de implementação e o início do fluxo de teste. Finalmente, o início da atividade *Planejar a Distribuição* marca o início do fluxo de distribuição e o término do

fluxo de teste. O término do processo marca o término do fluxo de distribuição. Suponha que deseja-se saber quando cada um dos fluxos de engenharia irá se iniciar ao mais cedo e se finalizar ao mais tarde, considerando o cenário Sc_2 . Para isso, basta extrair das árvores de prova canônica construídas para o cenário Sc_2 a data de início ao mais cedo da atividade que marca o início de um fluxo e a data de início ao mais tarde da atividade que marca o término deste mesmo fluxo (ou o início do fluxo posterior). A Tabela 7.1 mostra as datas de início ao mais cedo de cada um dos fluxos centrados na engenharia do processo RUP mapeados na t-Time WorkFlow net da Figura 7.4. A Tabela 7.2 mostra as datas de término ao mais tarde de cada um dos fluxos centrados na engenharia do processo RUP mapeados na t-Time WorkFlow net da Figura 7.4. Basta substituir as datas simbólicas das Tabelas 7.1 e 7.2 por datas numéricas de iterações distintas do processo RUP para saber quando cada um dos fluxos se iniciará e se finalizará.

Fluxo	Data de início ao mais cedo
Modelagem de Negócio	D_S
Requisitos	$D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}}$
Análise e Projeto	$D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}} + \max(d_{12_{min}}, d_{13_{min}} + d_{14_{min}} + d_{15_{min}} + d_{16_{min}} + d_{17_{min}}) + d_{18_{min}}$
Implementação	$D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}} + \max(d_{12_{min}}, d_{13_{min}} + d_{14_{min}} + d_{15_{min}} + d_{16_{min}} + d_{17_{min}}) + d_{18_{min}} + d_{19_{min}} + d_{20_{min}} + \max(d_{21_{min}}, d_{22_{min}} + d_{23_{min}} + \max(d_{24_{min}}, d_{25_{min}}) + d_{26_{min}}) + d_{27_{min}}$
Teste	$D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}} + \max(d_{12_{min}}, d_{13_{min}} + d_{14_{min}} + d_{15_{min}} + d_{16_{min}} + d_{17_{min}}) + d_{18_{min}} + d_{19_{min}} + d_{20_{min}} + \max(d_{21_{min}}, d_{22_{min}} + d_{23_{min}} + \max(d_{24_{min}}, d_{25_{min}}) + d_{26_{min}}) + d_{27_{min}} + d_{28_{min}} + d_{29_{min}} + d_{30_{min}} + d_{31_{min}} + d_{32_{min}}$
Distribuição	$D_S + d_{1_{min}} + d_{6_{min}} + d_{11_{min}} + \max(d_{12_{min}}, d_{13_{min}} + d_{14_{min}} + d_{15_{min}} + d_{16_{min}} + d_{17_{min}}) + d_{18_{min}} + d_{19_{min}} + d_{20_{min}} + \max(d_{21_{min}}, d_{22_{min}} + d_{23_{min}} + \max(d_{24_{min}}, d_{25_{min}}) + d_{26_{min}}) + d_{27_{min}} + d_{28_{min}} + d_{29_{min}} + d_{30_{min}} + d_{31_{min}} + d_{32_{min}} + d_{33_{min}} + d_{34_{min}} + d_{35_{min}} + d_{36_{min}} + \max(d_{38_{min}}, d_{37_{min}} + d_{39_{min}}) + d_{40_{min}}$

Tabela 7.1: Datas de início ao mais cedo para cada um dos fluxos de engenharia do processo RUP, considerando o cenário Sc_2 .

Fluxo	Data de término ao mais tarde
Modelagem de Negócio	$D_S + d_{1max} + d_{6max} + d_{11max}$
Requisitos	$D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max}$
Análise e Projeto	$D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max} + d_{19max} + d_{20max} + \max(d_{21max}, d_{22max} + d_{23max} + \max(d_{24max}, d_{25max}) + d_{26max}) + d_{27max}$
Implementação	$D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max} + d_{19max} + d_{20max} + \max(d_{21max}, d_{22max} + d_{23max} + \max(d_{24max}, d_{25max}) + d_{26max}) + d_{27max} + d_{28max} + d_{29max} + d_{30max} + d_{31max} + d_{32max}$
Teste	$D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max} + d_{19max} + d_{20max} + \max(d_{21max}, d_{22max} + d_{23max} + \max(d_{24max}, d_{25max}) + d_{26max}) + d_{27max} + d_{28max} + d_{29max} + d_{30max} + d_{31max} + d_{32max} + d_{33max} + d_{34max} + d_{35max} + d_{36max} + \max(d_{38max}, d_{37max} + d_{39max}) + d_{40max}$
Distribuição	$D_S + d_{1max} + d_{6max} + d_{11max} + \max(d_{12max}, d_{13max} + d_{14max} + d_{15max} + d_{16max} + d_{17max}) + d_{18max} + d_{19max} + d_{20max} + \max(d_{21max}, d_{22max} + d_{23max} + \max(d_{24max}, d_{25max}) + d_{26max}) + d_{27max} + d_{28max} + d_{29max} + d_{30max} + d_{31max} + d_{32max} + d_{33max} + d_{34max} + d_{35max} + d_{36max} + \max(d_{38max}, d_{37max} + d_{39max}) + d_{40max} + d_{41max} + d_{42max} + \max(d_{43max}, d_{44max}) + d_{45max} + d_{46max} + d_{47max}$

Tabela 7.2: Datas de término ao mais tarde para cada um dos fluxos de engenharia do processo RUP, considerando o cenário Sc_2 .

7.2 Considerações Finais do Capítulo

O presente capítulo mostrou a análise qualitativa e quantitativa de uma iteração do processo de Engenharia de Software denominado RUP, através da transformação de um Diagrama de Atividades da UML que especifica as atividades deste processo em uma Workflow net correspondente, para a qual foram realizadas as análises qualitativa e quantitativa.

Capítulo 8

Conclusão e Trabalhos Futuros

As contribuições de trabalho foram:

1. a definição de um método para a análise qualitativa e quantitativa de WorkFlow nets baseado nas árvores de prova canônica da lógica linear;
2. a definição de uma abordagem para a verificação e análise de especificações de processos de workflow em UML através da transformação de Diagramas de Atividades da UML em WorkFlow nets.

A análise qualitativa apresentada neste trabalho teve por objetivo provar o critério de corretude definido para as WorkFlow nets que é denominado *soundness*.

A análise quantitativa proposta neste trabalho baseou-se no cálculo de datas simbólicas para o planejamento de recursos utilizados na realização de cada tarefa do processo de workflow.

Para a verificação das especificações de processos de workflow especificados em Diagramas de Atividades da UML, foram apresentadas regras formais para transformar Diagramas de Atividades da UML em WorkFlow nets. Estas regras foram implementadas na linguagem de transformação ATL. Embora [Störrle et al. 2005] tenha proposto este tipo de transformação, o seu trabalho não apresentou um conjunto completo de transformações, nem mostrou a possibilidade da prova do critério de corretude *soundness* para a WorkFlow net gerada. Além disso, foi apresentada uma abordagem para encontrar e corrigir pontos críticos em Diagramas de Atividades da UML baseando-se nas árvores de prova canônica da lógica linear.

As vantagens das abordagens apresentadas são diversas. O fato de trabalhar com lógica linear permite provar o critério de corretude *soundness* em tempo linear e sem que seja necessária a construção de um grafo das marcações acessíveis, considerando diretamente a própria estrutura da WorkFlow net, ao invés de considerar seu autômato correspondente.

Além disso, o cálculo de datas simbólicas correspondentes à execução de cada tarefa mapeada em uma t-Time WorkFlow net permite planejar a utilização dos recursos envolvidos nas atividades do processo de workflow, através de fórmulas que podem ser

utilizadas por qualquer caso tratado pelo processo de workflow correspondente, sem que seja necessário percorrer novamente o processo de workflow inteiro para recalcular, para cada novo caso, datas de início e término das atividades envolvidas no processo.

A principal vantagem da abordagem apresentada para a verificação de processos de workflow especificados em Diagramas de Atividades da UML é a transformação de um modelo semi-formal em um modelo formal, para o qual algumas propriedades, como *soundness*, podem ser formalmente verificadas.

Como trabalho futuro, seria interessante adaptar ou desenvolver um provador de teoremas automático para a prova dos sequentes da lógica linear apresentados neste trabalho, de forma que este provador pudesse também calcular as datas de produção e consumo dos átomos presentes na árvore de prova.

Seria interessante também desenvolver um software que realizasse a tradução de uma WorkFlow net em fórmulas da lógica linear. Gerando, assim, os sequentes que poderiam ser provados automaticamente pelo provador de teoremas automático. Desta forma o processo de verificação da propriedade *soundness* e de cálculo de datas simbólicas para o planejamento de recursos ficaria totalmente automatizado.

Um outro trabalho interessante seria o de otimizar o software desenvolvido para a realização das transformações, de forma que o mesmo viesse a permitir como entrada um Diagrama de Atividades da UML gráfico e que mostrasse a WorkFlow net correspondente também graficamente.

Uma outra proposta interessante seria a de estender o metamodelo do Diagrama de Atividades da UML apresentado neste trabalho, acrescentando a noção de objetos (estruturas de dados), e propor uma transformação deste novo metamodelo em alguma rede de Petri de alto nível, como por exemplo as redes de Petri a objetos [Sibertin-Blanc 1985], permitindo, assim, a verificação de boas propriedades do modelo estendido.

São resultados do presente trabalho os artigos:

- [Soares Passos e Julia 2009a], apresentado no V Simpósio Brasileiro de Sistemas de Informação (SBSI 2009).
- [Soares Passos e Julia 2009b], aceito para publicação pela IEEE International Conference on Systems, Man, and Cybernetics (SMC 2009).
- [Soares Passos et al. 2009], submetido ao 12th Brazilian Symposium on Formal Methods (SBMF 2009).

Referências Bibliográficas

- [Atlas_Group] Atlas_Group. ATL: ATLAS Transformation Language. <http://www.eclipse.org/m2m/at1/>. Acesso em 27 de maio de 2009.
- [Bi e Zhao 2004] Bi, H. H. e Zhao, J. L. (2004). Applying Propositional Logic to Workflow Verification. *Information Technology and Management*, 5(3-4):293-318.
- [Gehrke et al. 1998] Gehrke, T., Goltz, U., e Wehrheim, H. (1998). The Dynamic Models of UML: Towards a Semantics and its Application in the Development Process. Technical report, Institut für Informatik, Universität Hildesheim.
- [Girard 1987] Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science*, 50(1):1-102.
- [Girard 1995] Girard, J.-Y. (1995). Linear logic: its syntax and semantics. In *Proceedings of the workshop on Advances in linear logic*, pp. 1-42, New York, NY, USA. Cambridge University Press.
- [Girault 1997] Girault, F. (1997). *A logic for Petri nets*, volume 31. Eddition Hermes.
- [Gochet e Gribomont 1990] Gochet, P. e Gribomont, P. (1990). *Logique: méthodes pour l'informatique fondamentale*, volume Vol 1. Hermès.
- [Jouault et al. 2008] Jouault, F., Allilaire, F., Bézivin, J., e Kurtev, I. (2008). ATL: A model transformation tool. *Science Computer Programming*, 72(1-2):31-39.
- [Julia e Soares 2003] Julia, S. e Soares, M. S. (2003). Verification of Real Time UML specifications through a specialized inference mechanism based on a Token Player Algorithm and the sequent calculus of Linear Logic. In *Proceedings of the 15th European Simulation Symposium and Exhibition*, pp. 65-70, Delft, The Netherlands.
- [Khansa 1997] Khansa, W. (1997). *Réseaux de Petri p-temporels contribution a l'etude des systemes a evenements discrets*. PhD thesis, Université de Savoie, France.
- [Kotb e Badreddin 2005] Kotb, Y. T. e Badreddin, E. (2005). Synchronization among Activities in a Workflow Using Extended Workflow Petri Nets. In *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pp. 548-551, Washington, DC, USA. IEEE Computer Society.
- [Kruchten 2003] Kruchten, P. (2003). *Introdução ao RUP: Rational Unified Process*. Editora Ciência Moderna, 2 edition.
- [Li e Song 2005] Li, S. e Song, B. (2005). Normalized workflow net (NWF-net): its definition and properties. *Future Gener. Comput. Syst.*, 21(7):1004-1014.

- [Lin e Qu 2004] Lin, C. e Qu, Y. (2004). Temporal inference of workflow systems based on time Petri nets: Quantitative and qualitative analysis. *Int. J. Intell. Syst.*, 19(5):417–442.
- [Ling e Schmidt 2000] Ling, S. e Schmidt, H. (2000). Time Petri nets for workflow modelling and analysis. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, pp. 3039–3044 vol.4.
- [Mantel e Otten 1999] Mantel, H. e Otten, J. (1999). linTAP: A Tableau Prover for Linear Logic. In *TABLEAUX '99: Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pp. 217–231, London, UK. Springer-Verlag.
- [Meena et al. 2005] Meena, H. K., Saha, I., Mondal, K. K., e Prabhakar, T. V. (2005). An approach to workflow modeling and analysis. In *Eclipse'05: Proceedings of the 2005 OOPSLA Workshop on Eclipse technology eXchange*, pp. 85–89, San Diego, California, New York, NY, USA. ACM.
- [Merlin 1974] Merlin, P. (1974). *A study of recoverability of computer systems*. PhD thesis, University of California, Irvine.
- [Murata 1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.
- [OMG 2008] OMG (2008). *OMG Unified Modeling Language Specification – Version 2.2*. Object Management Group. <http://www.omg.org/spec/UML/2.2/Superstructure/PDF>. Acesso em 27 de maio de 2009.
- [Petri 1962] Petri, C. A. (1962). *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn.
- [Pradin-Chezalviel et al. 1999] Pradin-Chezalviel, B., Valette, R., e Kunzle, L. A. (1999). Scenario Durations Characterization of T-Timed Petri Nets Using Linear Logic. In *PNPM '99: Proceedings of the The 8th International Workshop on Petri Nets and Performance Models*, p. 208, Washington, DC, USA. IEEE Computer Society.
- [Ramchandani 1974] Ramchandani, C. (1974). Analysis of Asynchronous Concurrent Systems by Timed Petri nets. Technical report, Cambridge, MA, USA.
- [Riviere et al. 2001] Riviere, N., Valette, R., Pradin-Chezalviel, B., e Ups, I. A. . (2001). Reachability and Temporal Conflicts in t-Time Petri Nets. In *PNPM '01: Proceedings of the 9th international Workshop on Petri Nets and Performance Models (PNPM'01)*, p. 229, Washington, DC, USA. IEEE Computer Society.
- [Ronan Champagnat 2000] Ronan Champagnat, Brigitte Pradin-Chézalviel, R. V. (2000). Petri nets and Linear Logic as an aid for scheduling batch processes. In *ADPM 2000, Automation of mixed processes: Hybrid Dynamic Systems*, pp. 107–112, Dortmund Germany.
- [Sibertin-Blanc 1985] Sibertin-Blanc, C. (1985). High Level Petri Nets with Data Structure. In Jensen, K. (editor), *Proceedings of the 6th european Workshop on Application and Theory of Petri Nets, Espoo, Finland*, pp. 141–170.

- [Sifakis 1977] Sifakis, J. (1977). Use of Petri Nets for Performance Evaluation. In *Proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems*, pp. 75–93, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.
- [Soares 2004] Soares, M. d. S. (2004). Uma abordagem baseada num jogador de redes de Petri p-temporal e no cálculo de sequentes da Lógica Linear para a verificação de cenários de Sistemas Tempo Real especificados através de diagramas dinâmicos da UML. Master's thesis, Faculdade de Computação, Universidade Federal de Uberlândia.
- [Soares Passos e Julia 2009a] Soares Passos, L. M. e Julia, S. (2009a). Análise qualitativa e quantitativa de WorkFlow nets utilizando Lógica Linear. In *SBSI 2009: Proceedings of the V Simpósio Brasileiro de Sistemas de Informação*, Brasília-DF, Brazil. SBC.
- [Soares Passos e Julia 2009b] Soares Passos, L. M. e Julia, S. (2009b). Qualitative Analysis of WorkFlow nets using Linear Logic: Soundness Verification. Submetido ao IEEE International Conference on Systems, Man, and Cybernetics - 2009.
- [Soares Passos et al. 2009] Soares Passos, L. M., Magalhães Júnior, T. A., Maia, M. A., e Julia, S. (2009). Verification of Workflow Specifications in UML Using Automated Transformations to WF-nets. Submetido ao 12th Brazilian Symposium on Formal Methods.
- [Störrle 2005] Störrle, H. (2005). Semantics and Verification of Data Flow in UML 2.0 Activities. *Electronic Notes in Theoretical Computer Science*, 127(4):35–52.
- [Störrle et al. 2005] Störrle, H., Hausmann, J. H., e Paderborn, U. (2005). Towards a formal semantics of UML 2.0 activities. In *Proceedings German Software Engineering Conference, volume P-64 of LNI*, pp. 117–128.
- [Tammet 1994] Tammet, T. (1994). Proof strategies in linear logic. *Journal of Automated Reasoning*, 12(3):273–304.
- [Tamura 1997] Tamura, N. (1997). A Linear Logic Prover (llprover). <http://bach.istc.kobe-u.ac.jp/llprover>. Acesso em 27 de maio de 2009.
- [Tričković 2000] Tričković, I. (2000). Formalizing Activity Diagram of UML by Petri nets. *Novi Sad Journal of Mathematics*, 30(3):161–171.
- [Valette 1979] Valette, R. (1979). Analysis of Petri Nets by Stepwise Refinements. *Journal of Computer and System Sciences*, 18:35–46.
- [Valette e Cardoso 1997] Valette, R. e Cardoso, J. (1997). *Redes de Petri*. Editora DAUSFC.
- [van der Aalst 1996] van der Aalst, W. M. P. (1996). Structural Characterizations of Sound Workflow Nets. *Computing Science Reports/23*, (96).
- [van der Aalst 1997] van der Aalst, W. M. P. (1997). Verification of Workflow Nets. In *ICATPN '97: Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pp. 407–426, London, UK. Springer-Verlag.

- [van der Aalst 1998] van der Aalst, W. M. P. (1998). The Application of Petri nets to Workflow Management. In *The Journal of Circuits, Systems and Computers*, pp. 21–66.
- [van der Aalst e van Hee 2004] van der Aalst, W. M. P. e van Hee, K. (2004). *Workflow Management: Models, Methods, and Systems*. The MIT Press.
- [Vilallonga et al. 2003] Vilallonga, G., Riesco, D., Montejano, G., e Uzal, R. (2003). Petri nets with clocks for the analytical validation of business process. pp. 11–25.

Apêndice: Código ATL para o mapeamento de Diagramas de Atividades da UML em WorkFlow nets

```

1  module ActivityDiagram2WorkflowNet;
2  create OUT : WorkflowNet from IN : ActivityDiagram;
3
4  helper def : globalInitialPlace : WorkflowNet!Place = OclUndefined;
5  helper def : globalFinalPlace : WorkflowNet!Place = OclUndefined;
6  helper def : globalWorkflowNet : WorkflowNet!WorkflowNet = OclUndefined;
7
8
9  rule ad2wn {
10     from ad : ActivityDiagram!Activity
11     to wn : WorkflowNet!WorkflowNet (
12     )
13
14     do {
15         thisModule.globalWorkflowNet <- wn;
16     }
17 }
18
19 rule Action2Transition {
20     from action : ActivityDiagram!Action
21
22     to transition : WorkflowNet!Transition (
23         name <- action.name,
24         net <- action.ad
25     )
26 }
27
28 rule DecisionNode2Place {
29     from decisionNode : ActivityDiagram!DecisionNode
30     to place : WorkflowNet!Place (
31         name <- decisionNode.name,
32         net <- decisionNode.ad
33     )
34 }
35
36 rule MergeNode2Place {
37     from mergeNode : ActivityDiagram!MergeNode
38     to place : WorkflowNet!Place (
39         name <- mergeNode.name,
40         net <- mergeNode.ad
41     )
42 }
43

```

```

44 rule ControlFlow {
45     from controlFlow : ActivityDiagram!ControlFlow (
46         controlFlow.source.oclType() <> ActivityDiagram!InitialNode and
47         controlFlow.target.oclType() <> ActivityDiagram!FinalNode and
48         controlFlow.target.oclType() <> ActivityDiagram!DecisionNode and
49         controlFlow.source.oclType() <> ActivityDiagram!DecisionNode and
50         controlFlow.target.oclType() <> ActivityDiagram!MergeNode and
51         controlFlow.source.oclType() <> ActivityDiagram!MergeNode)
52
53     to transitionToPlace : WorkflowNet!TransitionToPlace (
54         "from" <- controlFlow.source ,
55         net <- controlFlow.ad
56     ),
57
58     placeToTransition : WorkflowNet!PlaceToTransition (
59         "to" <- controlFlow.target ,
60         net <- controlFlow.ad
61
62     ), place : WorkflowNet!Place (
63         name <- ''
64     )
65
66     do {
67         transitionToPlace."to" <- place;
68         placeToTransition."from" <- place;
69
70         place.incomingEdge <- transitionToPlace;
71         place.outgoingEdge <- placeToTransition;
72         place.net <- thisModule.globalWorkflowNet;
73     }
74 }
75
76 rule ControlFlowFromDecisionMergeNodesToFinalNode {
77     from controlFlow : ActivityDiagram!ControlFlow (
78         (controlFlow.source.oclType() = ActivityDiagram!DecisionNode or
79         controlFlow.source.oclType() = ActivityDiagram!MergeNode) and
80         controlFlow.target.oclType() = ActivityDiagram!FinalNode)
81
82     to
83
84         placeToTransition : WorkflowNet!PlaceToTransition (
85             "from" <- controlFlow.source ,
86             net <- controlFlow.ad
87         ),
88
89         transitionToPlace : WorkflowNet!TransitionToPlace (
90             "to" <- controlFlow.target ,
91             net <- controlFlow.ad
92         ),
93
94         transition : WorkflowNet!Transition (
95             name <- 'aux'
96         )
97
98     do {
99         transitionToPlace."from" <- transition;
100         placeToTransition."to" <- transition;
101
102         transition.incomingEdge <- placeToTransition;
103         transition.outgoingEdge <- transitionToPlace;

```

```

103         transition.net <- thisModule.globalWorkflowNet;
104     }
105 }
106
107 rule ControlFlowFromInitialNodeToDecisionMergeNodes {
108     from controlFlow : ActivityDiagram!ControlFlow (
109         controlFlow.source.oclType() = ActivityDiagram!InitialNode and
110         (controlFlow.target.oclType() = ActivityDiagram!DecisionNode or
111         controlFlow.target.oclType() = ActivityDiagram!MergeNode))
112
113     to
114         placeToTransition : WorkflowNet!PlaceToTransition (
115             "from" <- controlFlow.source ,
116             net <- controlFlow.ad
117         ),
118
119         transitionToPlace : WorkflowNet!TransitionToPlace (
120             "to" <- controlFlow.target ,
121             net <- controlFlow.ad
122         ),
123
124         transition : WorkflowNet!Transition (
125             name <- 'aux'
126         )
127
128     do {
129         transitionToPlace."from" <- transition;
130         placeToTransition."to" <- transition;
131
132         transition.incomingEdge <- placeToTransition;
133         transition.outgoingEdge <- transitionToPlace;
134         transition.net <- thisModule.globalWorkflowNet;
135     }
136 }
137
138 rule ControlFlowFromDecisionMergeNodesToDecisionMergeNodes {
139     from controlFlow : ActivityDiagram!ControlFlow (
140         (controlFlow.source.oclType() = ActivityDiagram!DecisionNode and
141         controlFlow.target.oclType() = ActivityDiagram!DecisionNode)
142     or
143         (controlFlow.source.oclType() = ActivityDiagram!MergeNode and
144         controlFlow.target.oclType() = ActivityDiagram!MergeNode)
145     or
146         (controlFlow.source.oclType() = ActivityDiagram!MergeNode and
147         controlFlow.target.oclType() = ActivityDiagram!DecisionNode)
148     or
149         (controlFlow.source.oclType() = ActivityDiagram!DecisionNode and
150         controlFlow.target.oclType() = ActivityDiagram!MergeNode)
151     )
152
153     to
154         placeToTransition : WorkflowNet!PlaceToTransition (
155             "from" <- controlFlow.source ,
156             net <- controlFlow.ad
157         ),
158
159         transitionToPlace : WorkflowNet!TransitionToPlace (
160             "to" <- controlFlow.target ,
161             net <- controlFlow.ad

```

```

162         ),
163
164         transition : WorkflowNet!Transition (
165             name <- 'aux'
166         )
167
168     do {
169
170         transitionToPlace."from" <- transition;
171         placeToTransition."to" <- transition;
172
173         transition.incomingEdge <- placeToTransition;
174         transition.outgoingEdge <- transitionToPlace;
175         transition.net <- thisModule.globalWorkflowNet;
176     }
177 }
178
179
180 rule ForkNode2Transition {
181     from forkNode : ActivityDiagram!ForkNode
182
183     to transition : WorkflowNet!Transition (
184         name <- forkNode.name,
185         net <- forkNode.ad
186     )
187 }
188
189 rule JoinNode2Transition {
190     from joinNode : ActivityDiagram!JoinNode
191
192     to transition : WorkflowNet!Transition (
193         name <- joinNode.name,
194         net <- joinNode.ad
195     )
196 }
197
198 rule ControlFlowFromInitialNode {
199     from controlFlow : ActivityDiagram!ControlFlow (
200         controlFlow.source.ocltType() = ActivityDiagram!InitialNode and
201         controlFlow.target.ocltType() <> ActivityDiagram!DecisionNode and
202         controlFlow.target.ocltType() <> ActivityDiagram!MergeNode)
203
204     to placeToTransition : WorkflowNet!PlaceToTransition (
205         "from" <- controlFlow.source ,
206         "to" <- controlFlow.target ,
207         net <- controlFlow.ad
208     )
209 }
210
211 rule ControlFlowToFinalNode {
212     from controlFlow : ActivityDiagram!ControlFlow (
213         controlFlow.source.ocltType() <> ActivityDiagram!MergeNode and
214         controlFlow.source.ocltType() <> ActivityDiagram!DecisionNode and
215         controlFlow.target.ocltType() = ActivityDiagram!FinalNode)
216
217     to transitionToPlace : WorkflowNet!TransitionToPlace (
218         "from" <- controlFlow.source ,
219         "to" <- controlFlow.target ,
220         net <- controlFlow.ad

```

```

221         )
222     }
223
224     rule ControlFlowFromDecisionMergeNodes {
225         from controlFlow : ActivityDiagram!ControlFlow (
226             (controlFlow.source.ocltType() = ActivityDiagram!DecisionNode or
227              controlFlow.source.ocltType() = ActivityDiagram!MergeNode) and
228             controlFlow.target.ocltType() <> ActivityDiagram!FinalNode and
229             controlFlow.target.ocltType() <> ActivityDiagram!DecisionNode and
230             controlFlow.target.ocltType() <> ActivityDiagram!MergeNode)
231
232         to placeToTransition : WorkflowNet!PlaceToTransition (
233             "from" <- controlFlow.source ,
234             "to" <- controlFlow.target ,
235             net <- controlFlow.ad
236         )
237     }
238
239     rule ControlFlowToDecisionMergeNodes {
240         from controlFlow : ActivityDiagram!ControlFlow(
241             (controlFlow.target.ocltType() = ActivityDiagram!DecisionNode or
242              controlFlow.target.ocltType() = ActivityDiagram!MergeNode) and
243             controlFlow.source.ocltType() <> ActivityDiagram!InitialNode and
244             controlFlow.source.ocltType() <> ActivityDiagram!DecisionNode and
245             controlFlow.source.ocltType() <> ActivityDiagram!MergeNode)
246
247         to transitionToPlace : WorkflowNet!TransitionToPlace (
248             "from" <- controlFlow.source ,
249             "to" <- controlFlow.target ,
250             net <- controlFlow.ad
251         )
252     }
253
254     rule InitialNode2Place {
255         from initialNode : ActivityDiagram!InitialNode
256         to place : WorkflowNet!Place (
257             name <- initialNode.name,
258             net <- initialNode.ad
259         )
260
261         do {
262             thisModule.globalInitialPlace <- place;
263         }
264     }
265
266     rule FinalNode2Place {
267         from finalNode : ActivityDiagram!FinalNode
268         to place : WorkflowNet!Place (
269             name <- finalNode.name,
270             net <- finalNode.ad
271         )
272
273         do {
274             thisModule.globalFinalPlace <- place;
275         }
276     }

```