Uma Ferramenta para Mapear o Modelo Organizacional $\mathcal{M}$oise$^+$ em Rede de Petri Colorida de Forma Automatizada

Arthur da Silva Z. Cardoso[1], Ricardo A. Machado[1], Eder M. Gonçalves[1], Diana F. Adamatti[1]

[1]Centro de Ciências Computacionais - Universidade Federal do Rio Grande (FURG)
Rio Grande - RS - Brasil

Resumo. *Este artigo apresenta como proposta o mapeamento de modelos organizacionais do $\mathcal{M}$oise$^+$ para Redes de Petri Coloridas, focando na automatização desse serviço. Desta forma, a ideia é o desenvolvimento de uma ferramenta capaz de criar uma rede de Petri colorida a partir do arquivo XML do modelo $\mathcal{M}$oise$^+$ existente. A ferramenta está sendo desenvolvida em Python e já apresenta resultados preliminares.*

1. Introdução

Devido a falta de ferramentas especializadas para o desenvolvimento automatizado de uma rede de Petri colorida (RPC) utilizando CPN Tools, percebeu-se a necessidade de desenvolver uma ferramenta capaz de automatizar, pelo menos o processo de inicialização de uma rede, facilitando sua criação e no melhor dos casos podendo gerar uma rede inteira, apenas passando como entrada o XML do modelo $\mathcal{M}$oise$^+$ existente. Portanto, tem-se o objetivo de facilitar a interação com os modelos e suas criações, utilizando um algoritmo que possa gerar uma rede de Petri colorida, mesmo que seja em sua forma inicial, usando a linguagem de programação Python, sendo um parser que gerará arquivos para os elementos de um arquivo da ferramenta CPN Tools.

Uma Rede de Petri é um modelo abstrato e formal usado para descrever o fluxo de informação em sistemas, onde componentes dinâmicos são capturados por *tokens* que podem se mover entre estados, permitindo que comportamentos assíncronos e concorrentes sejam capturados [Haddad and Poitrenaud 1999]. Dado que os agentes são frequentemente solicitados a realizar várias tarefas simultaneamente, as redes de Petri são uma ferramenta particularmente eficaz para especificar e testar o comportamento de sistemas baseados em agentes. Além disso, as redes de Petri são um dos formalismos mais confiáveis para simulação e análise de modelagem comportamental e análise de falha de SMA [Boucherit et al. 2020] e são capazes de integrar os diferentes níveis e tipos da especificação $\mathcal{M}$oise$^+$, como resultado de seus mecanismos de abstração e refinamento hierárquico.

Por ser um trabalho em andamento, serão apresentadas algumas partes da implementação já realizadas e também os próximos passos da realização do trabalho. Acredita-se que esse um trabalho promissor, que tem a capacidade de ajudar com a forma como as informações são trabalhadas entre estas duas abordagens: $\mathcal{M}$oise$^+$ e CPN Tools. Desta forma, será mais simples tanto para quem for começar a utilizá-las, quanto para quem já a utiliza, de forma que sua criação seja mais simplificada e tenha-se uma melhor fluidez quando nos referimos a criação de novas redes de Petri.

2. Fundamentação Teórica

Existem atualmente diferentes modelos de organização para SMA, um deles é o $\mathcal{M}$oise$^+$, que implementa um modelo de programação para a dimensão organizacional de um sistema multiagente. Essa abordagem inclui uma linguagem para a organização do sistema e especificação de sua infraestrutura, com suporte para mecanismos de raciocínio baseados na organização no nível do agente. Esse modelo contém uma Especificação Estrutural (EE) e uma Especificação Funcional (EF) independentes, que estão ligadas por uma Especificação Deôntica (ED) [Hübner and Sichman 2007].

As Redes de Petri Coloridas (RPC) combinam as capacidades de uma rede de Petri ordinária com as capacidades de uma linguagem de programação. Nas RPC, as fichas possuem um valor de dados anexado que são chamados de cores. Essas cores podem representar diferentes processos ou recursos de uma rede, reduzindo assim o tamanho dos modelos. Para um determinado lugar, todas as fichas devem ter cores que pertençam a um tipo específico que é chamado de conjunto de cores do lugar. Esse uso de conjunto de cores é equivalente aos tipos de dados nas linguagens de programação (inteiro, caractere, booleano) e a especificação dos conjuntos de cores e das variáveis da rede é feita por declarações [Jensen 1997].

Em [Gonçalves et al. 2022] uma formalização de mapeamento para o Modelo $\mathcal{M}$oise$^+$ em RPC, denominado CPN4M foi apresentado. Nesse trabalho, diferentes elementos do $\mathcal{M}$oise$^+$ se transformam em estruturas para a RPC. Por exemplo, metas são representados pelos lugares; os papéis e grupos são relacionados através das cores; também foram levados em consideração os diferentes operadores de planos que podem ser uma sequência, uma escolha ou um paralelismo.

3. Arquitetura Desenvolvida

A Figura 1 mostra como será realizada essa automatização. Primeiramente, é realizada a leitura do arquivo XML do $\mathcal{M}$oise$^+$, utilizando um *Parser*, que é um componente de software que recebe dados de entrada em forma de texto e constrói uma estrutura de dados a partir dele.

A ferramenta, que utiliza a linguagem de programação *Python*, então identifica os elementos do $\mathcal{M}$oise$^+$, como os papéis e grupos que pertencem a parte da EE e as metas e operadores de planos usados na EF. A partir destes elementos, é possível gerar um novo arquivo que possui sua estrutura toda, também escrito em XML, porém sua extensão tem a sigla CPN, pois ele será lido pela ferramenta CPN Tools.

Para que esse arquivo funcione adequadamente, ele precisa conter os elementos da RPC. Portanto, devem ser criadas funções para gerar as declarações, os lugares, as transições e os arcos. Terminada essa etapa, o arquivo CPN é gerado e estará pronto para ser lido pelo CPN Tools.

O código começa com a importação do parser ao qual se irá utilizar, no caso dessas partes iniciais, só se fez necessário a utilização do XML ElementTree, que será usado em cada elemento das funções, como mostrado na Figura 2.

Em seguida parte-se para o desenvolvimento do lugar, que seria o equivalente a uma nova janela do CPN Tools, como mostrado na Figura 3 com o nome de wtitle e do

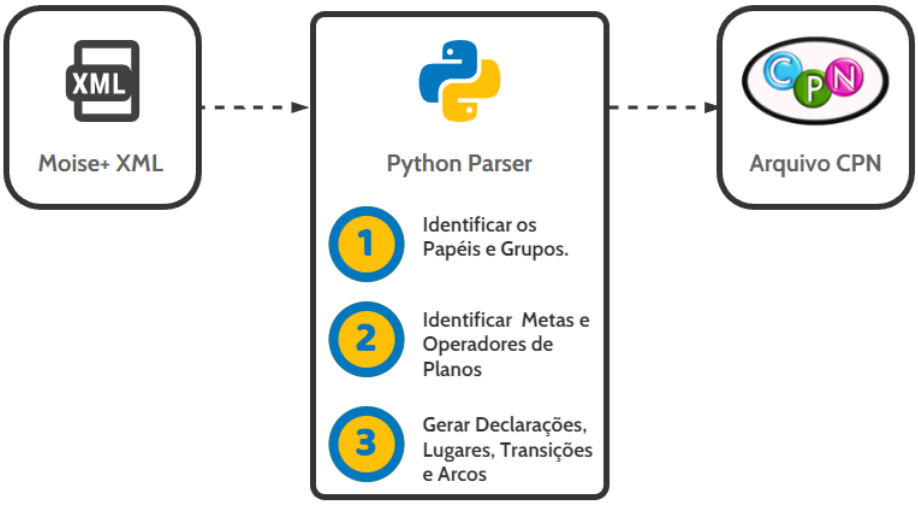

Figura 1. Visão Geral das Etapas de Automatização do Mapeamento

```
def gera_lugar(name, type):
  p1 = ET.SubElement(b37, "place")
  p1.set('id', 'ID1415407980')
  p2 = ET.SubElement(p1, "posattr")
  p2.set('x', '0.000000')
  p2.set('y', '-53.000000')
  p3 = ET.SubElement(p1, "fillattr")
  p3.set('colour', 'White')
  p3.set('pattern', '')
  p3.set('filled', 'false')
  p4 = ET.SubElement(p1, "lineattr")
  p4.set('colour', 'Black')
  p4.set('thick', '1')
  p4.set('type', 'solid')
  p5 = ET.SubElement(p1, "textattr")
  p5.set('colour', 'Black')
  p5.set('bold', 'false')
  p6 = ET.SubElement(p1, "text")
  p6.text = name
```

Figura 2. Exemplo da Criação de Lugar no Código de Arquitetura Desenvolvida

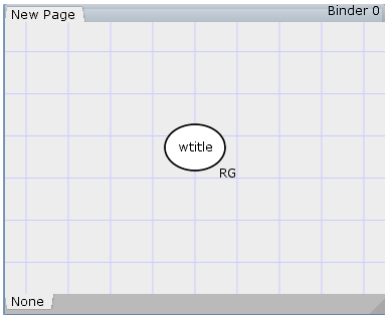

Figura 3. Exemplo da Criação de Lugar no CPN Tools

tipo RG. Por fim, tem-se o código que é gerado a partir desse algoritmo que gera o lugar, assim como está sendo mostrado na Figura 4.

4. Conclusões

Este trabalho apresentou uma abordagem para a criação de uma ferramenta que automatiza mapeamento organizacional de um SMA para uma RPC. Mais especificamente um modelo organizacional chamado $\mathcal{M}$oise$^+$ é convertido em elementos de RPC através de um método já descrito em trabalho anterior [Gonçalves et al. 2022]. Uma das possibilidades de utilização dessa ferramenta seria em ambientes de teste já que o modelo em RPC pode servir para para gerar os caminhos de teste necessários para a execução de testes.

Como próximos passos estão a conclusão desta ferramenta fazendo com que ela receba o arquivo XML do $\mathcal{M}$oise$^+$ como entrada, identifique os papéis, metas, a sequência com que as metas são realizadas, as relações deônticas entre as missões e os papéis, e as cardinalidades. Com essas informações já será possível gerar o XML da RPC completamente estruturada de acordo com a especificações $\mathcal{M}$oise$^+$ enviadas, utilizando as declarações, os lugares, transições e arcos da RPC.

Referências

Boucherit, A., Castro, L. M., Khababa, A., and Hasan, O. (2020). Petri net and rewriting logic based formal analysis of multi-agent based safety-critical systems. *Multiagent and Grid Systems*, 16(1):47–66.

Gonçalves, E. M. N., Machado, R. A., Rodrigues, B. C., and Adamatti, D. (2022). Cpn4m: Testing multi-agent systems under organizational model m oise+ using colored petri nets. *Applied Sciences*, 12(12):5857.

Haddad, S. and Poitrenaud, D. (1999). Theoretical aspects of recursive petri nets. In *International Conference on Application and Theory of Petri Nets*, pages 228–247. Springer.

Hübner, J. and Sichman, J. (2007). Developing organised multi-agent systems using the $Moise^+$ model: Programming issues at the system and agent levels. In *Int. J. Accounting, Auditing and Performance Evaluation*, pages 1–10.

Jensen, K. (1997). *Coloured Petri nets: basic concepts, analysis methods and practical use*, volume 1. Springer Science & Business Media.

```
<page id="ID6">
  <pageattr name="New Page" />
  <place id="ID1415407980">
    <posattr x="0.000000" y="-53.000000" />
    <fillattr colour="White" pattern="" filled="false" />
    <lineattr colour="Black" thick="1" type="solid" />
    <textattr colour="Black" bold="false" />
    <text>wtitle</text>
    <ellipse w="60.000000" h="46.000000" />
    <token x="-10.000000" y="0.000000" />
    <marking x="0.000000" y="0.000000" hidden="true">
      <snap snap_id="0" anchor.horizontal="0" anchor.vertical="0" />
    </marking>
    <type id="ID1415409755">
      <posattr x="32.500000" y="-79.000000" />
      <fillattr colour="White" pattern="Solid" filled="false" />
      <lineattr colour="Black" thick="0" type="Solid" />
      <textattr colour="Black" bold="false" />
      <text tool="CPN Tools" version="4.0.1">RG</text>
    </type>
    <initmark id="ID1415409615">
      <posattr x="56.000000" y="-26.000000" />
      <fillattr colour="White" pattern="Solid" filled="false" />
      <lineattr colour="Black" thick="0" type="Solid" />
      <textattr colour="Black" bold="false" />
      <text tool="CPN Tools" version="4.0.1" />
    </initmark>
  </place>
</page>
```

Figura 4. Exemplo da Criação de Lugar no Arquivo Gerado