

**VINÍCIUS LEOBET BREGOLI**

**DESENVOLVIMENTO DE GERADOR DE  
MODELOS DE SIMULAÇÃO PARA TOMADA  
DE DECISÃO NO CURTO PRAZO USANDO  
PROCESS MINING**

**Curitiba**

**2025**

**VINÍCIUS LEOBET BREGOLI**

**DESENVOLVIMENTO DE GERADOR DE MODELOS DE  
SIMULAÇÃO PARA TOMADA DE DECISÃO NO CURTO  
PRAZO USANDO PROCESS MINING**

Trabalho de Conclusão de Curso apresentado ao  
Curso de Engenharia de Computação da Pon-  
tifícia Universidade Católica do Paraná como  
requisito parcial para obtenção do grau de Ba-  
charel em Engenharia de Computação.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ – PUCPR  
ESCOLA POLITÉCNICA  
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Orientador: Prof. Dr. Edson Emílio Scalabrin

Curitiba  
2025

# Resumo

A tomada de decisão no curto prazo em ambientes complexos, como centros cirúrgicos, exige ferramentas capazes de integrar dados históricos e informações em tempo real. Nesse contexto, a mineração de processos (Process Mining – PM) e a simulação computacional emergem como tecnologias para apoiar gestores na alocação eficiente de recursos, detecção de desvios e previsão de cenários. Este trabalho propõe o desenvolvimento de um gerador de modelos de simulação baseado em PM, com foco em reduzir o esforço humano e o tempo necessário para a construção de modelos. O gerador utiliza logs de eventos como fonte de dados para criar automaticamente modelos de simulação, permitindo avaliar alternativas de curto prazo e apoiar a tomada de decisão operacional. O estudo se apoia no framework PM4SOS, estendido e adaptado para o domínio hospitalar, e integra métodos multicritério e técnicas de otimização. O resultado esperado é a disponibilização de um protótipo que auxilie gestores a analisar filas, prever ocupação de salas, otimizar agendas e reduzir gargalos, contribuindo para maior eficiência operacional, redução de custos e melhor qualidade no atendimento.

**Palavras-chave:** Mineração de Processos, Simulação Computacional, Tomada de Decisão, Otimização, Agendamento.

# Abstract

Short-term decision-making in complex environments, such as surgical centers, requires tools capable of integrating historical data and real-time information. In this context, Process Mining (PM) and computer simulation emerge as key technologies to support managers in efficient resource allocation, deviation detection, and scenario prediction. This work proposes the development of a simulation model generator based on PM, focusing on reducing human effort and the time required to build models. The generator uses event logs as a data source to automatically create simulation models, allowing the evaluation of short-term alternatives and supporting operational decision-making. The study is based on the PM4SOS framework, extended and adapted to the hospital domain, and integrates multicriteria methods and optimization techniques. The expected outcome is a prototype that helps managers analyze queues, predict room occupancy, optimize schedules, and reduce bottlenecks, contributing to greater operational efficiency, cost reduction, and improved service quality.

**Keywords:** Process Mining, Computer Simulation, Decision-Making, Optimization, Scheduling.

# **Lista de ilustrações**

Figura 1 – Exemplo de Rede de Petri representando um processo de negócio. Fonte: Autor (2025) . . . . .	23
Figura 2 – Fluxo completo da metodologia. Fonte: Autor (2025) . . . . .	36
Figura 3 – Fase 1: Análise Automática de Logs. Fonte: Autor (2025) . . . . .	37
Figura 4 – Fase 2: Mineração de Processos. Fonte: Autor (2025) . . . . .	42
Figura 5 – Fase 3: Simulação de Logs Sintéticos. Fonte: Autor (2025) . . . . .	47
Figura 6 – Fase 4: Validação de Qualidade. Fonte: Autor (2025) . . . . .	51
Figura 7 – Interface do usuário. Fonte: Autor (2025) . . . . .	60

# **Lista de tabelas**

Tabela 1 – Comparação entre algoritmos de descoberta de processos . . . . .	43
Tabela 2 – Parâmetros de configuração da simulação . . . . .	48
Tabela 3 – Interpretação de métricas de validação . . . . .	53
Tabela 4 – Parâmetros configuráveis do sistema . . . . .	53
Tabela 5 – Bibliotecas e ferramentas utilizadas . . . . .	54

# Sumário

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>10</b>
<b>1.1</b>	<b>Contexto e Problema . . . . .</b>	<b>11</b>
<b>1.2</b>	<b>Motivação . . . . .</b>	<b>13</b>
<b>1.3</b>	<b>Estado da Arte . . . . .</b>	<b>14</b>
<b>1.3.1</b>	<b>Mineração de Processos . . . . .</b>	<b>14</b>
<b>1.3.2</b>	<b>Simulação de Eventos Discretos . . . . .</b>	<b>15</b>
<b>1.3.3</b>	<b>Integração PM–DES . . . . .</b>	<b>15</b>
<b>1.3.4</b>	<b>Aplicações em domínios complexos . . . . .</b>	<b>15</b>
<b>1.3.5</b>	<b>Lacunas e oportunidades . . . . .</b>	<b>16</b>
<b>1.4</b>	<b>Soluções Similares . . . . .</b>	<b>16</b>
<b>1.5</b>	<b>Objetivos . . . . .</b>	<b>17</b>
<b>1.5.1</b>	<b>Objetivo Geral . . . . .</b>	<b>17</b>
<b>1.5.2</b>	<b>Objetivos Específicos . . . . .</b>	<b>17</b>
<b>1.6</b>	<b>Justificativa . . . . .</b>	<b>18</b>
<b>2</b>	<b>REFERENCIAL TEÓRICO . . . . .</b>	<b>20</b>
<b>2.1</b>	<b>Mineração de Processos . . . . .</b>	<b>20</b>
<b>2.2</b>	<b>Inductive Miner . . . . .</b>	<b>22</b>
<b>2.3</b>	<b>Redes de Petri . . . . .</b>	<b>22</b>
<b>2.4</b>	<b>Simulação de Eventos Discretos . . . . .</b>	<b>24</b>
<b>2.5</b>	<b>Análise Estatística . . . . .</b>	<b>26</b>
<b>2.6</b>	<b>Métricas de Qualidade de Modelos . . . . .</b>	<b>27</b>
<b>2.7</b>	<b>Geração de Logs Sintéticos . . . . .</b>	<b>28</b>
<b>2.8</b>	<b>Indicadores de Eficiência Operacional (ORE) . . . . .</b>	<b>30</b>
<b>2.8.1</b>	<b>Fundamentos do ORE . . . . .</b>	<b>30</b>
<b>2.8.2</b>	<b>Dimensões e Perdas Operacionais . . . . .</b>	<b>30</b>
<b>2.8.2.1</b>	<b>Perdas de Disponibilidade . . . . .</b>	<b>31</b>
<b>2.8.2.2</b>	<b>Perdas de Desempenho . . . . .</b>	<b>31</b>
<b>2.8.2.3</b>	<b>Perdas de Qualidade . . . . .</b>	<b>32</b>
<b>2.8.3</b>	<b>ORE como Driver de Simulação . . . . .</b>	<b>32</b>
<b>2.9</b>	<b>Padrão XES (eXtensible Event Stream) . . . . .</b>	<b>33</b>
<b>3</b>	<b>METODOLOGIA . . . . .</b>	<b>35</b>
<b>3.1</b>	<b>Visão Geral da Abordagem Metodológica . . . . .</b>	<b>35</b>
<b>3.2</b>	<b>Fluxo de Dados Detalhado . . . . .</b>	<b>35</b>

<b>3.3</b>	<b>Etapa 1: Análise Automática de Logs . . . . .</b>	<b>37</b>
<b>3.3.1</b>	<b>Diagrama de Componente . . . . .</b>	<b>37</b>
<b>3.3.2</b>	<b>Objetivo . . . . .</b>	<b>38</b>
<b>3.3.3</b>	<b>Justificativa . . . . .</b>	<b>38</b>
<b>3.3.4</b>	<b>Detecção de Atributos-Chave . . . . .</b>	<b>38</b>
<b>3.3.5</b>	<b>Coleta de Estatísticas Estruturais . . . . .</b>	<b>38</b>
<b>3.3.6</b>	<b>Análise de Eficiência Operacional (ORE) . . . . .</b>	<b>39</b>
<b>3.3.6.1</b>	<b>Cálculo de Métricas ORE . . . . .</b>	<b>39</b>
<b>3.3.7</b>	<b>Saída Integrada da Etapa . . . . .</b>	<b>40</b>
<b>3.4</b>	<b>Etapa 2: Mineração de Processos . . . . .</b>	<b>41</b>
<b>3.4.1</b>	<b>Diagrama de Componente . . . . .</b>	<b>41</b>
<b>3.4.2</b>	<b>Objetivo . . . . .</b>	<b>41</b>
<b>3.4.3</b>	<b>Filtragem de Variantes . . . . .</b>	<b>42</b>
<b>3.4.3.1</b>	<b>Problema . . . . .</b>	<b>42</b>
<b>3.4.3.2</b>	<b>Solução . . . . .</b>	<b>42</b>
<b>3.4.3.3</b>	<b>Parâmetro Padrão . . . . .</b>	<b>42</b>
<b>3.4.3.4</b>	<b>Justificativa . . . . .</b>	<b>42</b>
<b>3.4.4</b>	<b>Descoberta do Modelo de Processo . . . . .</b>	<b>42</b>
<b>3.4.4.1</b>	<b>Algoritmo Selecionado . . . . .</b>	<b>42</b>
<b>3.4.4.2</b>	<b>Justificativa da Escolha . . . . .</b>	<b>43</b>
<b>3.4.4.3</b>	<b>Saída . . . . .</b>	<b>43</b>
<b>3.4.5</b>	<b>Extração de Estatísticas Temporais . . . . .</b>	<b>43</b>
<b>3.4.5.1</b>	<b>Cálculo de Durações . . . . .</b>	<b>43</b>
<b>3.4.6</b>	<b>Ajuste de Distribuições Estatísticas . . . . .</b>	<b>44</b>
<b>3.4.6.1</b>	<b>Objetivo . . . . .</b>	<b>44</b>
<b>3.4.6.2</b>	<b>Distribuições Candidatas . . . . .</b>	<b>44</b>
<b>3.4.7</b>	<b>Avaliação de Qualidade do Modelo . . . . .</b>	<b>44</b>
<b>3.4.7.1</b>	<b>Fitness (0-1, maior é melhor) . . . . .</b>	<b>44</b>
<b>3.4.7.2</b>	<b>Precision (0-1, maior é melhor) . . . . .</b>	<b>45</b>
<b>3.4.7.3</b>	<b>Simplicity (0-1, maior é melhor) . . . . .</b>	<b>45</b>
<b>3.4.8</b>	<b>Saída da Etapa . . . . .</b>	<b>46</b>
<b>3.5</b>	<b>Etapa 3: Simulação de Logs Sintéticos . . . . .</b>	<b>46</b>
<b>3.5.1</b>	<b>Diagrama de Componente . . . . .</b>	<b>46</b>
<b>3.5.2</b>	<b>Objetivo . . . . .</b>	<b>48</b>
<b>3.5.3</b>	<b>Paradigma de Simulação . . . . .</b>	<b>48</b>
<b>3.5.3.1</b>	<b>Abordagem Escolhida . . . . .</b>	<b>48</b>
<b>3.5.3.2</b>	<b>Justificativa . . . . .</b>	<b>48</b>
<b>3.5.4</b>	<b>Configuração da Simulação . . . . .</b>	<b>48</b>
<b>3.5.5</b>	<b>Geração de Casos . . . . .</b>	<b>48</b>

<b>3.5.5.1</b>	<b>Processo de Chegadas . . . . .</b>	<b>48</b>
<b>3.5.6</b>	<b>Simulação Individual de Casos . . . . .</b>	<b>49</b>
<b>3.5.7</b>	<b>Geração dos Logs de Saída . . . . .</b>	<b>49</b>
<b>3.5.7.1</b>	<b>Formato Intermediário (CSV) . . . . .</b>	<b>49</b>
<b>3.5.7.2</b>	<b>Conversão para XES . . . . .</b>	<b>49</b>
<b>3.5.8</b>	<b>Saída da Etapa . . . . .</b>	<b>50</b>
<b>3.6</b>	<b>Etapa 4: Validação de Qualidade . . . . .</b>	<b>50</b>
<b>3.6.1</b>	<b>Diagrama de Componente . . . . .</b>	<b>50</b>
<b>3.6.2</b>	<b>Objetivo . . . . .</b>	<b>51</b>
<b>3.6.3</b>	<b>Métricas de Validação . . . . .</b>	<b>51</b>
<b>3.6.3.1</b>	<b>Similaridade Estrutural (Fitness de Alinhamento) . . . . .</b>	<b>51</b>
<b>3.6.3.2</b>	<b>Similaridade de Distribuição de Atividades . . . . .</b>	<b>52</b>
<b>3.6.3.3</b>	<b>Similaridade de Distribuição de Variantes . . . . .</b>	<b>52</b>
<b>3.6.3.4</b>	<b>Similaridade de Distribuição de Durações . . . . .</b>	<b>52</b>
<b>3.6.3.5</b>	<b>Validação Estatística (Teste de Kolmogorov-Smirnov) . . . . .</b>	<b>52</b>
<b>3.6.4</b>	<b>Agregação e Interpretação de Resultados . . . . .</b>	<b>53</b>
<b>3.6.5</b>	<b>Saída da Etapa . . . . .</b>	<b>53</b>
<b>3.7</b>	<b>Parâmetros e Configurações . . . . .</b>	<b>53</b>
<b>3.7.1</b>	<b>Tabela de Parâmetros Principais . . . . .</b>	<b>53</b>
<b>3.8</b>	<b>Ferramentas e Tecnologias . . . . .</b>	<b>54</b>
<b>3.8.1</b>	<b>Bibliotecas Principais . . . . .</b>	<b>54</b>
<b>3.8.2</b>	<b>Justificativa das Escolhas . . . . .</b>	<b>54</b>
<b>3.9</b>	<b>Pseudocódigo de Alto Nível . . . . .</b>	<b>54</b>
<b>4</b>	<b>DESENVOLVIMENTO . . . . .</b>	<b>56</b>
<b>4.1</b>	<b>Estrutura do Projeto . . . . .</b>	<b>56</b>
<b>4.1.1</b>	<b>Organização dos Módulos . . . . .</b>	<b>57</b>
<b>4.2</b>	<b>Arquitetura do Sistema . . . . .</b>	<b>57</b>
<b>4.3</b>	<b>Implementação dos Componentes Principais . . . . .</b>	<b>57</b>
<b>4.3.1</b>	<b>Módulo de Análise de Logs . . . . .</b>	<b>57</b>
<b>4.3.2</b>	<b>Módulo de Mineração de Processos . . . . .</b>	<b>57</b>
<b>4.3.3</b>	<b>Módulo de Simulação . . . . .</b>	<b>58</b>
<b>4.3.4</b>	<b>Módulo de Validação . . . . .</b>	<b>58</b>
<b>4.4</b>	<b>Implementação da Interface de Usuário . . . . .</b>	<b>59</b>
<b>4.4.1</b>	<b>Tecnologias Utilizadas . . . . .</b>	<b>59</b>
<b>4.5</b>	<b>Testes e Validação . . . . .</b>	<b>61</b>
<b>4.5.1</b>	<b>Teste de Análise Automática de Logs . . . . .</b>	<b>61</b>
<b>4.5.2</b>	<b>Teste de Mineração de Processos . . . . .</b>	<b>61</b>
<b>4.5.3</b>	<b>Teste de Simulação de Logs Sintéticos . . . . .</b>	<b>61</b>

<b>4.5.4</b>	<b>Teste de Validação de Qualidade</b>	<b>61</b>
<b>4.5.5</b>	<b>Teste de Integração Completa</b>	<b>62</b>
<b>4.5.6</b>	<b>Teste de Interface de Usuário</b>	<b>62</b>
<b>4.6</b>	<b>Resultados Esperados e Obtidos</b>	<b>62</b>
<b>4.6.1</b>	<b>Resultados Esperados</b>	<b>62</b>
<b>4.6.2</b>	<b>Resultados Obtidos</b>	<b>63</b>
<b>4.6.3</b>	<b>Limitações Identificadas</b>	<b>63</b>
<b>5</b>	<b>CONCLUSÃO</b>	<b>64</b>
<b>5.1</b>	<b>Limitações e Desafios</b>	<b>64</b>
<b>5.2</b>	<b>Considerações Finais</b>	<b>64</b>
	<b>REFERÊNCIAS</b>	<b>65</b>

# 1 Introdução

Em organizações modernas, a complexidade operacional dos processos exige decisões cada vez mais rápidas baseadas em dados. Setores como saúde, manufatura, logística, mineração e serviços compartilham desafios semelhantes: alocação eficiente de recursos, detecção de gargalos, redução de custos e melhoria contínua de desempenho. Em todos esses contextos, as decisões de curto prazo — aquelas que precisam ser tomadas em horizontes de horas ou dias — exercem impacto direto na produtividade, na utilização de recursos e na qualidade do serviço prestado.

Apesar da ampla digitalização de processos e do grande volume de dados coletados em sistemas corporativos, a conversão dessas informações em conhecimento útil para a tomada de decisão ainda depende, em grande parte, da análise manual e da experiência de gestores e especialistas. Esse processo, além de demorado, está sujeito a vieses cognitivos e erros humanos, o que limita a capacidade das organizações de reagir rapidamente a mudanças operacionais.

A mineração de processos (*Process Mining – PM*) surge como uma abordagem capaz de extrair, a partir de logs de eventos, informações estruturadas sobre o comportamento real dos processos. Essa técnica permite descobrir modelos de processo, identificar gargalos e analisar conformidade com base em dados reais de execução. Trabalhos recentes demonstram seu potencial em contextos industriais complexos, como sistemas logísticos internos ([WUENNENBERG; WEGERICHE; FOTTNER, 2023](#)) e processos de mineração subterrânea ([BRZYCHCZY; ZUBER; AALST, 2024](#)), nos quais a análise de eventos de sensores e sistemas de controle tem permitido identificar padrões de desempenho e oportunidades de otimização.

Entretanto, a mineração de processos, por si só, fornece uma visão descritiva do comportamento passado, não sendo suficiente para antecipar cenários futuros ou testar alternativas operacionais. Nesse sentido, a integração com a simulação computacional oferece uma perspectiva complementar, permitindo representar dinamicamente o sistema e avaliar o impacto de diferentes estratégias antes de sua aplicação real ([MARUŞTER; BEEST, 2009](#)). Modelos de simulação baseados em dados de execução viabilizam a previsão de tempos de espera, taxas de ocupação e desempenho global do processo, contribuindo para decisões mais assertivas.

Ainda assim, a construção manual de modelos de simulação é uma tarefa intensiva, que exige conhecimento técnico detalhado sobre o processo e considerável esforço de modelagem — fatores que inviabilizam seu uso em situações que demandam resposta rápida. Pesquisas recentes têm buscado reduzir essa lacuna, utilizando mineração de processos para automatizar a geração de modelos de simulação, como proposto por ([FERRONATO, 2022](#)) no framework PM4SOS. Essa abordagem demonstrou ser eficaz para o suporte operacional em centros cirúrgicos, combinando mineração de logs, simulação e otimização multicritério. De forma semelhante, estudos na área

de logística e engenharia de minas têm evidenciado os benefícios de abordagens híbridas entre mineração de dados e simulação discreta, com o objetivo de otimizar fluxos físicos e prever anomalias operacionais ([MENG et al., 2024](#)).

Neste contexto, o presente trabalho propõe o desenvolvimento de um **gerador de modelos de simulação para tomada de decisão no curto prazo**, fundamentado na integração entre *Process Mining* e *Simulação de Eventos Discretos*. O objetivo é automatizar a criação de modelos a partir de dados reais de execução, reduzindo o esforço cognitivo do analista e possibilitando a análise preditiva de cenários com base em evidências empíricas. Este projeto busca generalizar o conceito, tornando o gerador aplicável a diferentes domínios organizacionais que disponham de registros de eventos estruturados, utilizando o setor hospitalar como estudo de caso para validação prática.

Com essa proposta, pretende-se preencher lacunas identificadas na literatura e na prática organizacional, especialmente no que diz respeito à integração automatizada entre dados históricos e modelos de simulação. A pesquisa almeja demonstrar que a combinação de mineração de processos, modelagem automatizada e simulação orientada por dados constitui uma ferramenta eficaz para apoiar a **tomada de decisão operacional de curto prazo**, promovendo análises rápidas, reproduutíveis e sustentadas por dados reais.

## 1.1 Contexto e Problema

A crescente complexidade dos ambientes organizacionais e o dinamismo dos processos produtivos exigem das instituições uma capacidade contínua de adaptação e resposta rápida a mudanças operacionais. Setores como saúde, manufatura, logística, mineração e serviços compartilham desafios recorrentes: alocar recursos de forma eficiente, detectar gargalos, reduzir custos e promover a melhoria contínua de desempenho. Em todos esses contextos, as decisões de curto prazo — aquelas que precisam ser tomadas em intervalos de horas ou dias — exercem influência direta sobre a produtividade, a eficiência operacional e a qualidade do serviço prestado.

Com a intensificação da digitalização e a adoção de sistemas de informação integrados, grandes volumes de dados passaram a ser gerados em tempo real. Contudo, a transformação desses dados em conhecimento útil para apoiar decisões ainda depende, em grande medida, da experiência humana e de análises manuais. Esse processo é sujeito a vieses cognitivos e à limitação do tempo de resposta, o que reduz a capacidade organizacional de reagir a variações de demanda, atrasos ou falhas no fluxo produtivo.

Nesse cenário, a *mineração de processos* (*Process Mining – PM*) tem se consolidado como uma abordagem poderosa para extrair, a partir de logs de eventos, informações estruturadas sobre o comportamento real dos processos. A técnica permite descobrir modelos de processo, identificar desvios de conformidade e mensurar indicadores de desempenho com base em dados

reais de execução. Aplicações recentes demonstram seu potencial em domínios industriais complexos, como sistemas logísticos internos (WUENNENBERG; WEGERICHE; FOTTNER, 2023) e processos de mineração subterrânea (BRZYCHCZY; ŽUBER; AALST, 2024), nos quais a análise de dados sensoriais e transacionais tem contribuído para o diagnóstico e a otimização de operações.

Apesar disso, a mineração de processos oferece predominantemente uma visão descritiva e diagnóstica — concentrada no passado e no presente das operações. Por não possuir mecanismos preditivos, ela é limitada quando se busca antecipar cenários futuros, testar alternativas operacionais ou estimar o impacto de decisões sob diferentes condições de carga ou recursos.

Por outro lado, a *simulação de eventos discretos* (*Discrete-Event Simulation – DES*) é amplamente utilizada como ferramenta preditiva e experimental, permitindo avaliar o comportamento de sistemas sob múltiplos cenários e medir o impacto de alterações no fluxo de processos, políticas de recursos ou parâmetros operacionais. A integração entre simulação e mineração de processos tem sido explorada em diversas pesquisas, com destaque para (MARUŞTER; BEEST, 2009), que propõe um método de redesenho de processos baseado em modelos descobertos via PM e simulados em ferramentas de Petri Nets. De forma semelhante, estudos mais recentes demonstram a aplicação conjunta dessas técnicas em sistemas de logística e manufatura, utilizando simulação como meio de validar hipóteses e otimizar o desempenho global (WUENNENBERG; WEGERICHE; FOTTNER, 2023; MENG et al., 2024).

Entretanto, a construção manual de modelos de simulação ainda é um processo intensivo, que requer tempo, conhecimento técnico especializado e compreensão detalhada dos fluxos operacionais. Essa complexidade torna inviável o uso da simulação como instrumento cotidiano de apoio à decisão, especialmente em contextos que demandam reações rápidas a eventos inesperados. O desafio, portanto, está em como automatizar a geração de modelos de simulação a partir de dados reais, reduzindo o esforço de modelagem e ampliando a aplicabilidade da técnica no suporte à decisão operacional de curto prazo.

Trabalhos como o de (FERRONATO, 2022) propuseram soluções integradas baseadas em *Process Mining*, simulação e otimização multicritério, aplicadas ao contexto hospitalar para o agendamento de cirurgias. O framework PM4SOS demonstrou que a combinação dessas abordagens permite reduzir tempos de espera, ajustar a alocação de recursos e reagir dinamicamente a variações de demanda. A partir dessa base, observa-se a oportunidade de generalizar o conceito, tornando o mecanismo de geração automática de modelos aplicável a diferentes domínios — desde linhas de produção industriais até processos administrativos e logísticos.

Dessa forma, o problema central abordado nesta pesquisa pode ser formulado da seguinte maneira: como automatizar a criação de modelos de simulação baseados em dados reais, extraídos por mineração de processos, de modo a apoiar a tomada de decisão operacional em curto prazo?

A pesquisa busca preencher lacunas identificadas na literatura e na prática organizacional,

especialmente no que tange à integração automatizada entre dados históricos e modelos de simulação, reduzindo o esforço cognitivo do tomador de decisão e permitindo análises preditivas em tempo reduzido. O desenvolvimento de um gerador de modelos automatizado visa, portanto, aproximar o potencial analítico da mineração de processos da capacidade preditiva da simulação computacional, promovendo uma abordagem prática, escalável e orientada por dados para o apoio à decisão operacional.

## 1.2 Motivação

A transformação digital tem impulsionado a geração de grandes volumes de dados operacionais em praticamente todos os setores organizacionais. Esses dados, provenientes de sistemas corporativos, sensores e plataformas transacionais, representam uma fonte estratégica de informação sobre o comportamento real dos processos. No entanto, a maior parte desse potencial permanece subutilizada: os dados são frequentemente empregados apenas em análises descritivas, voltadas ao monitoramento retrospectivo, sem oferecer suporte efetivo à tomada de decisão em tempo hábil. Essa limitação é especialmente crítica em contextos de alta variabilidade, nos quais decisões de curto prazo precisam ser tomadas com base em evidências confiáveis e atualizadas.

Nesse cenário, a *mineração de processos* (*Process Mining – PM*) desponta como uma tecnologia promissora para a extração de conhecimento a partir de logs de eventos, permitindo compreender, auditar e aprimorar processos com base em dados reais. A PM tem sido aplicada com sucesso em domínios industriais complexos, como logística interna, manufatura e mineração (WUENNENBERG; WEGERICH; FOTTNER, 2023; BRZYCHCZY; ŻUBER; AALST, 2024), oferecendo diagnósticos precisos sobre gargalos, desvios e desempenho operacional. No entanto, sua natureza essencialmente descritiva ainda limita seu uso como instrumento de previsão ou de apoio dinâmico à decisão.

Por outro lado, a *simulação de eventos discretos* (*Discrete-Event Simulation – DES*) é amplamente reconhecida como uma ferramenta analítica capaz de explorar cenários alternativos e estimar o impacto de decisões antes de sua implementação. A combinação entre mineração de processos e simulação tem se mostrado particularmente poderosa, pois permite unir a observação empírica dos dados à experimentação virtual dos processos (MARUŞTER; BEEST, 2009). Entretanto, a etapa de construção de modelos de simulação ainda representa um obstáculo significativo, demandando esforço cognitivo elevado e conhecimento técnico especializado em modelagem e parametrização de sistemas.

A motivação central deste trabalho surge, portanto, da necessidade de automatizar a geração de modelos de simulação a partir de informações extraídas por mineração de processos. Essa automatização tem potencial para reduzir drasticamente o tempo e o esforço envolvidos na criação de modelos analíticos, ao mesmo tempo em que democratiza o acesso de gestores e analistas a ferramentas de apoio à decisão em ambientes complexos e dinâmicos. Além disso,

possibilita o uso de dados históricos e em tempo real como base para análises preditivas e prescritivas, elevando o nível de maturidade analítica das organizações.

Inspirado em iniciativas como o framework PM4SOS proposto por ([FERRONATO, 2022](#)), que integrou mineração de processos, simulação e otimização multicritério para o agendamento cirúrgico, o presente trabalho busca expandir esse conceito para diferentes domínios organizacionais. A proposta é criar um **gerador de modelos de simulação** com capacidade generalizável, aplicável a qualquer processo que possua logs de eventos estruturados, mantendo o ambiente hospitalar apenas como estudo de caso para validação experimental.

Ao promover a integração entre mineração de processos, simulação e otimização, esta pesquisa visa contribuir para o avanço das práticas de gestão operacional baseada em dados. Acredita-se que o desenvolvimento de uma ferramenta automatizada, capaz de gerar modelos de simulação em tempo reduzido, possa fortalecer o suporte à decisão no curto prazo, ampliando a eficiência, a agilidade e a capacidade de adaptação das organizações a ambientes cada vez mais dinâmicos e complexos.

## 1.3 Estado da Arte

Esta seção sintetiza os principais avanços nas áreas de *Process Mining* (PM), *Simulação de Eventos Discretos* (DES) e sua integração, destacando aplicações em domínios complexos (logística interna, manufatura, mineração e saúde) e, por fim, delineando as lacunas que motivam a presente pesquisa.

### 1.3.1 Mineração de Processos

A mineração de processos consolida-se como abordagem para extrair, a partir de logs de eventos, o comportamento real dos processos, contemplando tarefas de *descoberta* (derivação de modelos), *conformidade* (comparação de logs com modelos de referência) e *aprimoramento* (análise de desempenho e gargalos). Em ambientes industriais, a PM tem avançado no uso de dados transacionais e de sensores para diagnosticar desvios e medir indicadores operacionais ([WUENNENBERG; WEGERICHS; FOTTNER, 2023](#); [BRZYCHCZY; ŻUBER; AALST, 2024](#)). Um desafio recorrente em cenários físicos (ex.: mineração subterrânea) é a preparação do *event log*: identificação de *case ID*, abstração de eventos de baixo nível e tratamento de ruído; para isso, têm-se proposto estratégias de abstração supervisionadas e não supervisionadas, além de heurísticas específicas para correlação de eventos ([BRZYCHCZY; ŻUBER; AALST, 2024](#)). Em domínios administrativos e de P&D, a PM também tem sido explorada para gerir projetos, com o uso de algoritmos como o *Heuristics Miner* para descobrir fluxos e dependências ([JOE et al., 2018](#)).

### 1.3.2 Simulação de Eventos Discretos

A DES é uma técnica consolidada para previsão e análise *what-if*, permitindo estimar efeitos de políticas operacionais (alocação de recursos, regras de priorização) antes de sua implementação. Em gestão de operações, a simulação suporta a avaliação de throughput, tempos de espera, utilização e confiabilidade. Em sistemas físico-cibernéticos (ex.: logística e mineração), combinar dados de operação com simulação tem se mostrado essencial para entender efeitos de variabilidade e restrições de recursos (WUENNENBERG; WEGERICH; FOTTNER, 2023; MENG et al., 2024).

### 1.3.3 Integração PM–DES

A literatura propõe a integração entre PM e DES de forma a: (i) descobrir modelos a partir de dados reais; (ii) parametrizar e simular cenários no *As-Is* e *To-Be*; e (iii) comparar ganhos de desempenho (MARUŞTER; BEEST, 2009). Esse fluxo permite que modelos descobertos via PM sejam transformados em modelos simuláveis (e.g., Petri nets/CPN) para avaliar alternativas de redesenho e impacto em indicadores. Mais recentemente, frameworks em logística interna descrevem um *pipeline* end-to-end que parte do sistema real, gera dados sintéticos por DES quando necessário, transforma saídas de simulação em *event logs* e realiza descoberta/conformidade para suportar a otimização iterativa (WUENNENBERG; WEGERICH; FOTTNER, 2023). Em mineração, a integração de dados sensoriais, PM e simulação tem sido aplicada para explicar ciclos operacionais, localizar gargalos e fundamentar ações de melhoria, inclusive com técnicas de abstração de eventos e análise comparativa (BRZYCHCZY; ŽUBER; AALST, 2024). Em paralelo, abordagens de simulação modular têm ampliado a capacidade preditiva em fenômenos acoplados (mecânico–hidráulicos), ilustrando a necessidade de arquiteturas de acoplamento explícito e troca de dados entre módulos (MENG et al., 2024).

### 1.3.4 Aplicações em domínios complexos

Trabalhos recentes mostram metodologias que combinam PM e DES para diagnosticar gargalos, avaliar conformidade e otimizar parâmetros locais e globais de sistemas de fluxo de materiais, com iterações guiadas por KPIs (tempo de atravessamento, utilização, throughput) (WUENNENBERG; WEGERICH; FOTTNER, 2023). Além disso, ressalta-se a geração de *event logs* a partir de saídas de simulação para fechar o ciclo de descoberta/conformidade e acelerar o aprendizado sobre o sistema.

A PM tem sido empregada também para modelar processos como o ciclo de corte em *longwall*, enfrentando problemas de granularidade, ruído e case correlation; soluções combinam heurísticas e aprendizagem (supervisionada e não supervisionada) para identificação de atividades e instâncias, habilitando descoberta e análise de desempenho com dados de sensores (BRZYCHCZY; ŽUBER; AALST, 2024). Em paralelo, simulações acopladas (mecânica–escoamento)

têm ampliado a previsão de efeitos operacionais e riscos, reforçando o papel de modelos preditivos conectados a dados reais (MENG et al., 2024).

No contexto hospitalar, o PM4SOS integra PM, simulação e otimização multicritério para suporte operacional (e.g., agendamento cirúrgico) (FERRONATO, 2022). Em gestão de projetos, a PM tem servido à descoberta de fluxos e análise de desempenho, com foco em dependências, variações e papéis organizacionais (JOE et al., 2018).

### 1.3.5 Lacunas e oportunidades

Apesar do progresso, persistem lacunas relevantes:

- Muitas abordagens dependem de etapas manuais (abstração de eventos, mapeamento semântico de atividades, parametrização de tempos e recursos) para viabilizar a simulação; faltam ferramentas que automatizem a transformação de *event logs* em modelos simuláveis com parametrização consistente e reproduzível
- Soluções existentes são, em geral, específicas de domínio (hospitalar, logística, mineração), com limitações de portabilidade de *mapeamentos* e estruturas de dados
- A correlação de eventos, a abstração de sinais contínuos e a ainda demandam estratégias robustas e padronizadas para produção de *event logs* adequados à simulação.
- Há carência de pipelines que fechem o ciclo (dados → PM → geração automática de modelo → DES → recomendações/otimização) com tempos de processamento compatíveis com decisões operacionais de curto prazo.

Essas lacunas motivam o desenvolvimento de um *gerador de modelos de simulação* orientado por PM, com escopo generalizável e foco em decisões de curto prazo, reduzindo intervenções manuais, padronizando a parametrização e aproximando diagnóstico descritivo de validação preditiva.

## 1.4 Soluções Similares

Diversas pesquisas têm buscado integrar *Process Mining* (PM) e *Simulação de Eventos Discretos* (DES) como forma de aprimorar a compreensão e a predição do comportamento dos processos reais. Entretanto, a maioria das soluções existentes mantém um alto grau de dependência de intervenção manual, especialmente nas etapas de modelagem, parametrização e calibração.

Entre as iniciativas mais influentes, destaca-se a metodologia proposta por (MARUŞTER; BEEST, 2009), que combina mineração de processos e simulação para o redesenho organizacional. O método parte de logs reais para gerar modelos *As-Is*, simulá-los e compará-los com

versões otimizadas *To-Be*, permitindo estimar ganhos de desempenho. Apesar de pioneiro, o processo de conversão dos modelos minerados em modelos simuláveis requer ajustes manuais e conhecimento técnico em modelagem formal (como redes de Petri coloridas).

No contexto hospitalar, o framework PM4SOS, desenvolvido por (FERRONATO, 2022), integra mineração de processos, simulação e otimização multicritério para o agendamento cirúrgico. Essa abordagem automatiza parcialmente a geração de modelos e utiliza indicadores de eficiência para suportar decisões em tempo reduzido. Contudo, sua aplicação ainda é restrita ao domínio da saúde, carecendo de generalização para outros tipos de processos.

Na área industrial, trabalhos como (WUENNENBERG; WEGERICHE; FOTTNER, 2023) propõem pipelines que unem simulação e mineração de processos em sistemas logísticos internos. Esses modelos exploram o uso de simulação para geração de dados sintéticos, que são posteriormente minerados para verificação de conformidade e detecção de gargalos. Em paralelo, pesquisas em mineração subterrânea (BRZYCHCZY; ŻUBER; AALST, 2024) e simulação modular (MENG et al., 2024) também avançam na integração entre dados de sensores, abstração de eventos e análise preditiva, embora com foco em contextos físicos específicos.

Em síntese, as soluções atuais demonstram o potencial da integração entre PM e DES, mas permanecem limitadas quanto à automação de ponta a ponta e à adaptabilidade entre diferentes domínios. O presente trabalho propõe evoluir essas abordagens por meio de um gerador de modelos de simulação automatizado e generalizável, reduzindo o esforço técnico necessário e ampliando o alcance da análise preditiva em processos de decisão operacional de curto prazo.

## 1.5 Objetivos

### 1.5.1 Objetivo Geral

Desenvolver um **gerador de modelos de simulação baseado em mineração de processos** para apoiar a **tomada de decisão no curto prazo**, capaz de criar automaticamente modelos de simulação a partir de logs de eventos, reduzindo o esforço humano e o tempo necessário para a modelagem de sistemas complexos.

### 1.5.2 Objetivos Específicos

Para alcançar o objetivo geral, este trabalho busca atender aos seguintes objetivos específicos:

- Investigar métodos e técnicas de integração entre mineração de processos, simulação computacional e otimização multicritério;

- Projetar uma arquitetura de sistema capaz de gerar automaticamente modelos de simulação a partir de logs de eventos processados por ferramentas de PM;
- Implementar um protótipo funcional do gerador de modelos, com base em bibliotecas de mineração de processos e simulação (como PM4PY e SimPy);
- Aplicar o protótipo desenvolvido em um estudo de caso no contexto hospitalar, validando sua eficácia na geração de modelos e apoio à decisão operacional;
- Avaliar o desempenho do sistema proposto quanto à precisão dos modelos gerados, tempo de execução e potencial de generalização para outros domínios.

## 1.6 Justificativa

A crescente disponibilidade de dados operacionais e o avanço das tecnologias analíticas têm impulsionado o desenvolvimento de soluções voltadas à gestão baseada em evidências. No entanto, a transformação desses dados em modelos preditivos e prescritivos ainda depende, em grande parte, de atividades manuais de análise e modelagem, o que limita sua aplicabilidade em contextos que demandam decisões rápidas e precisas. Nesse cenário, a integração entre *mineração de processos* (PM) e *simulação de eventos discretos* (DES) surge como uma abordagem promissora para reduzir a distância entre o conhecimento descritivo e a previsão operacional.

Do ponto de vista **científico**, a pesquisa se justifica pela necessidade de ampliar o corpo de conhecimento existente sobre a integração entre PM e DES, especialmente no que se refere à automação das etapas de modelagem e parametrização. Trabalhos anteriores, como os de ([MARUŞTER; BEEST, 2009](#)) e ([WUENNENBERG; WEGERICHE; FOTTNER, 2023](#)), demonstraram o potencial dessa integração para diagnóstico e otimização de processos, mas ainda requerem intervenções manuais para gerar modelos simuláveis. Assim, o presente estudo contribui para o avanço teórico ao propor um método automatizado que amplia a reprodutibilidade e a aplicabilidade da simulação orientada por dados reais.

Sob a perspectiva **tecnológica**, a proposta apresenta relevância por desenvolver um **gerador automatizado de modelos de simulação**, capaz de transformar *event logs* em modelos DES de forma padronizada e escalável. Essa automação representa um avanço em relação a abordagens anteriores, como o PM4SOS ([FERRONATO, 2022](#)), que, embora integre PM e simulação, mantém dependência de ajustes manuais e é restrito a um domínio específico (o hospitalar). A ferramenta proposta busca ser generalizável e adaptável, permitindo sua aplicação em diferentes setores — como logística, manufatura e mineração — sem perda de precisão analítica.

Por fim, do ponto de vista **prático**, o trabalho se justifica pela crescente necessidade de apoiar decisões operacionais de curto prazo em ambientes complexos e dinâmicos. Organizações

modernas demandam respostas rápidas a variações de demanda, falhas operacionais e restrições de recursos, e a geração automatizada de modelos de simulação oferece uma alternativa eficiente para análise de cenários em tempo reduzido. A validação do gerador no contexto hospitalar reforça sua aplicabilidade real, demonstrando o potencial de transferência tecnológica da solução para outros domínios.

Dessa forma, o presente trabalho se justifica por unir relevância teórica, tecnológica e prática, propondo uma abordagem inovadora e generalizável para a automatização da geração de modelos de simulação baseados em mineração de processos. Espera-se, com isso, contribuir para a consolidação de um novo paradigma de *Process Mining* aplicado à tomada de decisão operacional orientada por dados, fortalecendo a integração entre análise descritiva, simulação preditiva e otimização de desempenho em tempo hábil.

## 2 Referencial Teórico

Este capítulo apresenta os fundamentos conceituais e técnicos que sustentam o desenvolvimento de um gerador de modelos de simulação orientado por mineração de processos (*Process Mining*). São abordados desde os conceitos fundamentais de mineração de processos até a integração com simulação de eventos discretos e indicadores de desempenho operacional aplicados a ambientes hospitalares.

### 2.1 Mineração de Processos

A mineração de processos (*Process Mining*) é uma disciplina que une conceitos de mineração de dados (*data mining*) e de gerenciamento de processos de negócio (*Business Process Management — BPM*) com o objetivo de extrair conhecimento útil a partir de registros de eventos (*event logs*) provenientes de sistemas de informação. Segundo van der Aalst ([AALST, 2016](#)), a mineração de processos tem como propósito descobrir, monitorar e aprimorar processos reais com base nos dados efetivamente registrados, constituindo uma ponte entre a análise orientada a dados e a modelagem formal de processos.

Com a crescente digitalização das operações empresariais e o avanço de sistemas como ERPs, CRMs e sistemas de controle de manufatura, passou a ser possível registrar detalhadamente cada etapa executada em um processo. Esses registros, conhecidos como *event logs*, formam a base para a aplicação de técnicas de mineração de processos. Cada evento registrado representa a execução de uma atividade pertencente a um caso (*case*) e contém atributos como o nome da atividade, o identificador do caso, o responsável (*resource*) e o carimbo de tempo (*timestamp*). A estrutura desses logs permite a reconstituição da sequência de atividades e o estudo do comportamento do processo real ([MARUŞTER; BEEST, 2009](#)).

A mineração de processos é composta por três grandes categorias de técnicas ([AALST, 2016](#)):

1. **Descoberta de processos (*Process Discovery*)**: tem como objetivo gerar automaticamente um modelo de processo a partir de um log de eventos, sem conhecimento prévio do fluxo. O modelo resultante pode ser representado em diferentes notações, como Redes de Petri, BPMN (*Business Process Model and Notation*) ou Árvores de Processo (*Process Trees*).
2. **Verificação de conformidade (*Conformance Checking*)**: consiste em comparar um modelo de processo pré-existente com um log de eventos real, avaliando a aderência entre o comportamento observado e o comportamento esperado. Essa comparação fornece métricas como *fitness* e *precision*.

3. **Aprimoramento de modelos (*Enhancement*)**: busca enriquecer modelos existentes com informações adicionais provenientes dos logs, como tempos médios de execução, gargalos, desvios e uso de recursos, permitindo a análise de desempenho e a detecção de oportunidades de otimização.

O ciclo de mineração de processos, descrito no *Process Mining Manifesto* da IEEE Task Force on Process Mining ([AALST, 2016](#)), enfatiza que a aplicação prática da técnica depende da disponibilidade e da qualidade dos logs de eventos. Logs incompletos, inconsistentes ou mal formatados comprometem a acurácia dos modelos descobertos. Nesse sentido, Kherbouche et al. ([KHERBOUCHE; LAGA; MASSE, 2020](#)) destacam a importância da avaliação da qualidade dos logs, propondo métricas de *completude*, *consistência* e *complexidade* antes da aplicação das técnicas de mineração.

Em termos de arquitetura, um sistema de mineração de processos segue um fluxo básico: coleta de dados, pré-processamento, mineração propriamente dita e análise dos resultados. Durante a coleta, os logs podem ser extraídos de sistemas como SAP, Oracle, ou de bancos de dados customizados. O pré-processamento envolve a limpeza e padronização dos dados, incluindo a identificação correta de casos e atividades. Em seguida, algoritmos como *Alpha Miner*, *Heuristic Miner* e *Inductive Miner* são aplicados para a descoberta do modelo de processo ([LEEMANS; FAHLAND; AALST, 2013](#)).

Os resultados podem ser apresentados em notações formais, como Redes de Petri, ou em linguagens mais visuais, como BPMN. O uso de ferramentas especializadas, como o *ProM Framework* e a biblioteca Python PM4Py, facilita essa análise e integração com outros métodos, como a simulação de eventos discretos ([FERRONATO; SCALABRIN, 2021](#)).

O padrão *eXtensible Event Stream* (XES), definido pela IEEE ([IEEE Computational Intelligence Society, 2010](#)), estabelece a estrutura de logs de eventos para garantir interoperabilidade entre ferramentas e consistência na troca de dados. Cada log em XES é composto por uma coleção de *traces* (casos), e cada *trace* contém uma sequência ordenada de *events*. Essa padronização é essencial para o sucesso de frameworks modernos de mineração de processos e simulação integrada.

Portanto, a mineração de processos se consolida como uma abordagem essencial para compreender, auditar e melhorar processos organizacionais em ambientes baseados em dados. Quando combinada com simulação e análise estatística, ela se torna uma ferramenta poderosa para suporte à decisão, otimização operacional e melhoria contínua de processos complexos, como os hospitalares e logísticos.

## 2.2 Inductive Miner

Entre os diversos algoritmos de descoberta de processos disponíveis, o *Inductive Miner* (IM), proposto por Leemans, Fahland e van der Aalst (LEEMANS; FAHLAND; AALST, 2013), é um dos mais relevantes e amplamente utilizados na literatura e em ferramentas modernas de mineração, como o *ProM* e o *PM4Py*.

Diferentemente de abordagens anteriores, como o *Alpha Miner* e o *Heuristic Miner*, o Inductive Miner foi projetado para garantir propriedades formais no modelo resultante, como a *soundness* (correção comportamental) e a estrutura hierárquica em blocos (*block-structured*). Essas propriedades asseguram que o modelo possa ser executado sem estados mortos ou impasses, além de facilitar sua conversão em linguagens formais como Redes de Petri e Árvores de Processo (*Process Trees*).

O princípio fundamental do algoritmo baseia-se na decomposição recursiva do log de eventos. O IM analisa as relações de precedência e causalidade entre atividades e divide o log em sublogs coerentes, representando fragmentos independentes do processo. Cada sublog é então minerado de forma isolada, e os resultados são combinados em uma estrutura hierárquica que reflete a composição lógica das atividades.

Uma das vantagens práticas do Inductive Miner é sua compatibilidade direta com representações de simulação. Ao produzir modelos formalmente corretos e livres de inconsistências estruturais, o IM facilita a transformação automática em modelos de simulação baseados em Redes de Petri, como destacado por Feronato (FERRONATO; SCALABRIN, 2021). Essa característica é essencial para o desenvolvimento de sistemas como o *PM2Sim*, que automatiza a criação de modelos de simulação de eventos discretos.

## 2.3 Redes de Petri

As Redes de Petri constituem um formalismo matemático e gráfico amplamente utilizado para modelar, analisar e simular sistemas de eventos discretos. Propostas originalmente por Carl Adam Petri na década de 1960 e formalizadas por Peterson (PETERSON, 1981), essas redes oferecem uma base rigorosa para a representação de processos dinâmicos caracterizados por concorrência, sincronização, conflito e causalidade — propriedades típicas de sistemas produtivos, logísticos e hospitalares.

Uma Rede de Petri é definida formalmente como uma tupla  $N = (P, T, F)$ , onde:

- $P$  representa o conjunto de lugares (*places*);
- $T$  representa o conjunto de transições (*transitions*);
- $F \subseteq (P \times T) \cup (T \times P)$  é o conjunto de arcos que conectam lugares e transições.

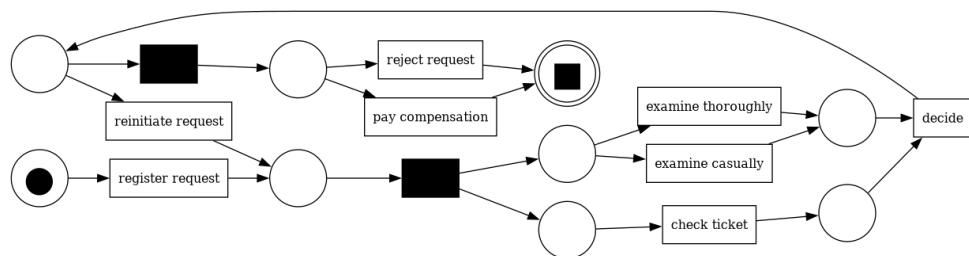
Os lugares simbolizam condições ou estados do sistema, enquanto as transições representam eventos ou atividades que modificam esses estados. A dinâmica do sistema é descrita por meio da movimentação de fichas (*tokens*) entre os lugares. O conjunto de tokens em um dado instante define a marcação atual da rede (*marking*), representando o estado do processo. Quando todas as condições de disparo de uma transição são satisfeitas, ela se torna habilitada e pode ser executada, consumindo e produzindo tokens conforme o fluxo definido em  $F$ . Essa semântica de disparo possibilita a modelagem de sistemas paralelos e assíncronos de forma intuitiva e precisa.

Segundo Peterson ([PETERSON, 1981](#)), uma das principais vantagens das Redes de Petri é a possibilidade de realizar análises formais de propriedades do sistema modelado, como:

- **Alcançabilidade (Reachability)**: determina quais estados podem ser atingidos a partir da marcação inicial.
- **Viveza (Liveness)**: garante que nenhuma transição se torne permanentemente inativa, evitando estados mortos.
- **Conservação (Boundedness)**: assegura que o número de tokens em cada lugar permanece finito, prevenindo explosões de estados.
- **Deadlock-freedom**: certifica que o sistema não entra em impasse completo.

Essas propriedades são fundamentais para a verificação de correção comportamental (*soundness*) em modelos de processos descobertos via mineração, assegurando que o modelo é executável e que todo caso pode ser concluído corretamente ([AALST, 2016](#)).

No contexto da mineração de processos, as Redes de Petri são amplamente utilizadas como formalismo intermediário para representar os modelos extraídos de logs de eventos. O *Inductive Miner*, por exemplo, gera diretamente uma Rede de Petri *sound* e *block-structured* ([LEEMANS; FAHLAND; AALST, 2013](#)), permitindo tanto a verificação de conformidade quanto a execução simulada do processo. Essa característica é essencial para a integração entre mineração e simulação de eventos discretos, pois possibilita a tradução direta de modelos minerados em estruturas simuláveis.



Na Figura 1, os elementos da Rede de Petri são representados da seguinte forma: os círculos representam *lugares* (places), que indicam estados ou condições do processo; os retângulos representam *transições* (transitions), que correspondem a atividades ou eventos que podem ocorrer; os círculos com pontos pretos internos representam *tokens* (marcas), que indicam a presença de uma condição ou o estado atual do processo; e os círculos com contorno destacado e tokens internos representam lugares marcados, indicando estados ativos no momento atual da execução.

Ferronato ([FERRONATO; SCALABRIN, 2021](#)) destaca que as Redes de Petri desempenham um papel central na integração entre mineração e simulação. Em seu framework *PM2Sim*, as redes extraídas a partir de logs de eventos são automaticamente convertidas em modelos de simulação de eventos discretos implementados em Python. Essa conversão permite avaliar métricas como tempo de ciclo, gargalos e utilização de recursos, transformando os modelos minerados em instrumentos de suporte à decisão operacional.

Em síntese, as Redes de Petri constituem uma linguagem formal robusta e expressiva para representar o comportamento de sistemas reais. Sua adoção no contexto da mineração de processos garante não apenas a fidelidade comportamental dos modelos gerados, mas também sua viabilidade para análises de desempenho e simulação, consolidando-se como elo fundamental entre a teoria de processos e a prática da modelagem computacional.

## 2.4 Simulação de Eventos Discretos

A simulação de eventos discretos (*Discrete Event Simulation — DES*) é uma técnica de modelagem computacional amplamente utilizada para representar sistemas dinâmicos em que o estado do sistema muda apenas em pontos discretos no tempo. Cada mudança é provocada por um evento, que ocorre em um instante específico e representa uma transição de estado. Essa abordagem é amplamente empregada em áreas como manufatura, logística, saúde e serviços, onde os processos são compostos por atividades sequenciais, paralelas e dependentes de recursos.

A DES permite reproduzir o comportamento de sistemas complexos sem a necessidade de interferir em seu ambiente real, possibilitando análises de desempenho, previsão de gargalos e avaliação de cenários alternativos. Diferentemente da simulação contínua, que modela fenômenos em tempo contínuo por meio de equações diferenciais, a simulação discreta descreve processos orientados por eventos, sendo, portanto, ideal para a representação de fluxos de trabalho e processos empresariais.

Em termos formais, um modelo de simulação de eventos discretos é composto por três elementos básicos:

- **Entidades:** representam os objetos que transitam pelo sistema (por exemplo, pacientes, ordens de serviço ou produtos).

- **Recursos:** correspondem aos elementos que executam as atividades (como profissionais, máquinas ou salas cirúrgicas).
- **Eventos:** são as ocorrências que alteram o estado do sistema, como o início ou o término de uma atividade.

Esses componentes são orquestrados por um *motor de simulação* (*simulation engine*) que mantém um relógio lógico e agenda os próximos eventos a serem processados. Cada evento executado pode gerar novos eventos futuros, modificando o estado do sistema e permitindo a evolução temporal do modelo (LIU, 2015).

A integração entre simulação e mineração de processos vem sendo explorada nos últimos anos. Liu (LIU, 2015) demonstrou que os modelos extraídos via *process mining* podem ser automaticamente convertidos em modelos de simulação de eventos discretos, reduzindo o esforço de modelagem e aumentando a precisão analítica. Essa integração é especialmente relevante em contextos onde a dinâmica do sistema se altera frequentemente, exigindo atualizações rápidas de modelos e previsões.

Nesse contexto, Ferronato e Scalabrin (FERRONATO; SCALABRIN, 2021) desenvolveram o framework *PM2Sim*, que automatiza a criação de modelos de simulação a partir de logs de eventos reais. O sistema identifica atividades, durações, tempos de espera e recursos envolvidos, transformando essas informações em um modelo DES implementado em Python com a biblioteca SimPy. Essa biblioteca fornece uma estrutura orientada a processos, baseada em geradores, que permite modelar entidades como processos que interagem em um ambiente temporal discreto. A SimPy oferece ainda suporte à criação de filas, controle de recursos, eventos simultâneos e coleta de estatísticas de desempenho, tornando-a ideal para aplicações em mineração de processos e análise operacional.

Além disso, o framework *PM2Sim* utiliza distribuições estatísticas ajustadas por meio de técnicas de *fitting* (adequação), com base em bibliotecas como NumPy e SciPy, permitindo representar o comportamento dos tempos de execução das atividades. O resultado é um modelo de simulação que reflete o comportamento observado nos logs, possibilitando a análise de métricas como tempo de ciclo, utilização de recursos e identificação de gargalos.

De acordo com Wuennenberg et al. (WUENNENBERG; WEGERICHE; FOTTNER, 2023), a combinação de mineração de processos e simulação discreta é particularmente útil em ambientes industriais e logísticos, onde a qualidade dos dados e a variabilidade operacional representam desafios significativos. Nesse sentido, a DES serve como ferramenta de validação e experimentação para modelos minerados, permitindo testar hipóteses e prever o impacto de mudanças estruturais antes de sua implementação no ambiente real.

## 2.5 Análise Estatística

A etapa de análise estatística é fundamental para a construção de modelos de simulação realistas e coerentes com os processos observados. Após a descoberta do modelo de processo e a extração das informações de desempenho a partir dos logs de eventos, é necessário realizar o ajuste estatístico dos parâmetros temporais, como durações de atividades, tempos de espera e intervalos entre eventos. Esses parâmetros são essenciais para que o modelo de simulação reflita adequadamente a variabilidade e o comportamento estocástico do sistema.

Segundo van der Aalst ([AALST, 2016](#)), a mineração de processos deve ser vista como uma disciplina orientada por dados (*data-driven*), e o sucesso de suas aplicações depende da correta interpretação das distribuições de tempo, frequência e desempenho que emergem dos registros de eventos. O uso de técnicas estatísticas complementa a fase de descoberta, permitindo transformar modelos descritivos em modelos quantitativos capazes de realizar simulações e análises preditivas.

Ferronato e Scalabrin ([FERRONATO; SCALABRIN, 2021](#)) destacam que o uso de distribuições estatísticas é um dos pilares do framework *PM2Sim*, que automatiza a criação de modelos de simulação a partir de logs reais. No sistema proposto, os tempos de execução das atividades e os intervalos entre eventos são ajustados a partir de funções de probabilidade clássicas, como Normal, Log-Normal e Exponencial. Essa parametrização é obtida por meio do ajuste de distribuições (*distribution fitting*), realizado com base nos dados históricos de cada atividade registrada no log. O processo envolve a estimativa dos parâmetros das distribuições e a verificação de aderência dos dados observados, assegurando a representatividade do modelo.

Para realizar esses ajustes, bibliotecas estatísticas como NumPy e SciPy são utilizadas no ambiente Python, permitindo estimar as distribuições de probabilidade e calcular medidas como média, variância, desvio padrão e coeficiente de variação. Além disso, testes de aderência, como o de Kolmogorov–Smirnov, são empregados para validar se os dados amostrais seguem adequadamente a distribuição teórica selecionada. Esse procedimento garante que os tempos de simulação não apenas reproduzam as médias históricas, mas também capturem a variabilidade natural do processo ([LIU, 2015](#)).

O resultado da análise estatística é incorporado diretamente aos parâmetros do modelo de simulação, alimentando o mecanismo de eventos discretos com tempos amostrados de distribuições probabilísticas. Essa abordagem permite representar o comportamento dinâmico e imprevisível dos processos reais, algo essencial em ambientes sujeitos a variações operacionais, como hospitais e sistemas logísticos. Conforme observado por Leemans et al. ([LEEMANS; FAHLAND; AALST, 2013](#)), a precisão dos tempos e a correta modelagem da frequência das atividades impactam diretamente a capacidade do modelo minerado em reproduzir o fluxo real dos eventos.

Assim, a análise estatística atua como uma ponte entre a descoberta de processos e

a simulação de eventos discretos, traduzindo dados empíricos em parâmetros quantitativos para o modelo. Essa integração garante que a simulação gerada preserve as características estatísticas do processo original, permitindo a avaliação confiável de métricas de desempenho e a experimentação de cenários alternativos de operação.

## 2.6 Métricas de Qualidade de Modelos

A qualidade dos modelos descobertos por meio da mineração de processos é um fator determinante para a confiabilidade das análises subsequentes e, principalmente, para o uso desses modelos como base para simulações. A avaliação sistemática da qualidade garante que o modelo minerado não apenas represente corretamente o comportamento histórico, mas também seja capaz de generalizar o comportamento futuro e sustentar decisões operacionais baseadas em evidências.

De acordo com van der Aalst ([AALST, 2016](#)), um modelo de processo deve ser avaliado em múltiplas dimensões de qualidade, de modo a equilibrar precisão, generalização e simplicidade. As principais métricas utilizadas na literatura são: *fitness*, *precision*, *generalization* e *simplicity*. Essas métricas, amplamente adotadas em ferramentas como o *ProM Framework* e a *PM4Py*, compõem a base dos algoritmos de verificação de conformidade (*conformance checking*), responsáveis por comparar o comportamento observado nos logs de eventos com o comportamento previsto pelo modelo minerado.

- **Fitness:** avalia o quanto o modelo reproduz o comportamento observado no log. Um modelo com alto *fitness* consegue reproduzir todas as sequências válidas de atividades sem apresentar falhas de execução.
- **Precision:** mede o nível de restrição do modelo, penalizando comportamentos não observados nos dados. Modelos excessivamente permissivos tendem a apresentar alta flexibilidade, mas baixa precisão.
- **Generalization:** representa a capacidade do modelo de capturar variações plausíveis do processo, evitando o sobreajuste aos dados históricos.
- **Simplicity:** reflete o grau de complexidade estrutural do modelo; modelos excessivamente complexos, embora precisos, dificultam a interpretação e a manutenção ([AALST et al., 2012](#)).

O *Process Mining Manifesto* ([AALST et al., 2012](#)) destaca a importância de equilibrar essas dimensões, pois a busca por um modelo com *fitness* perfeito pode levar à perda de generalização, e vice-versa. A maturidade da área de mineração de processos se reflete justamente na capacidade de lidar com esse compromisso entre precisão e abstração. Esse equilíbrio é

particularmente relevante quando os modelos descobertos serão empregados em simulações, uma vez que comportamentos não observados ou superajustados podem comprometer a validade dos resultados simulados.

Liu ([LIU, 2015](#)) ressalta que a integração entre mineração e simulação exige não apenas um modelo logicamente consistente, mas também estatisticamente representativo. A validação entre logs reais e logs sintéticos simulados permite medir a aderência entre ambos, utilizando métricas de distância e similaridade, como a *edit distance*. Essa abordagem garante que a simulação não apenas reproduza o fluxo de atividades, mas também mantenha coerência temporal e probabilística com o processo original.

Kherbouche et al. ([KHERBOUCHE; LAGA; MASSE, 2020](#)) complementam essa perspectiva ao argumentar que a qualidade do modelo está diretamente relacionada à qualidade do log de eventos. Logs incompletos, redundantes ou ruidosos geram modelos com baixa precisão e menor confiabilidade. Por isso, o controle da qualidade dos logs — avaliado por dimensões como *completude, consistência, acurácia e complexidade* — é pré-requisito para a obtenção de métricas significativas de *fitness* e *precision*.

Portanto, as métricas de qualidade de modelos constituem um elo crítico entre mineração e simulação. Elas fornecem os indicadores necessários para validar a representatividade, robustez e aplicabilidade do modelo descoberto. Um modelo de processo só é efetivamente útil quando combina boa aderência aos dados históricos com capacidade de generalização para novos cenários — característica essencial para o uso em ambientes de tomada de decisão e otimização operacional.

## 2.7 Geração de Logs Sintéticos

A geração de logs sintéticos é uma etapa estratégica na integração entre mineração de processos e simulação de eventos discretos. Essa técnica permite criar registros artificiais de eventos que preservam as propriedades estruturais e estatísticas de logs reais, viabilizando experimentos controlados, testes de desempenho e validação de modelos sem a necessidade de utilizar dados sensíveis ou restritos. Em termos práticos, os logs sintéticos servem como uma representação simulada do comportamento observado, permitindo avaliar a fidelidade dos modelos minerados e a confiabilidade das previsões operacionais.

Segundo van der Aalst ([AALST, 2016](#)), a utilização de logs artificiais é essencial para verificar se o modelo de processo descoberto é capaz de reproduzir o comportamento do sistema real. Essa comparação é comumente feita por meio de técnicas de *conformance checking*, nas quais o log sintético gerado pelo modelo é confrontado com o log original, medindo-se métricas como *fitness* e *precision*. Essa abordagem é particularmente útil em ambientes dinâmicos, onde a estrutura do processo pode mudar com frequência, como na área hospitalar e em sistemas

logísticos.

Ferronato ([FERRONATO; SCALABRIN, 2021](#)) propõe no framework *PM2Sim* um processo automatizado de geração de logs sintéticos a partir de modelos descobertos via mineração de processos. O sistema transforma o modelo minerado — geralmente representado como uma Rede de Petri — em um modelo de simulação implementado em Python por meio da biblioteca *SimPy*. Durante a execução da simulação, eventos são registrados em formato XES (*eXtensible Event Stream*), mantendo a compatibilidade com ferramentas de mineração como PM4Py e ProM. Essa estratégia permite a retroalimentação do ciclo de mineração, criando uma integração contínua entre descoberta, simulação e validação.

Augusto et al. ([AUGUSTO et al., 2016](#)) reforçam essa importância ao aplicar a integração entre mineração de processos e simulação em um contexto clínico. Os autores desenvolveram uma metodologia para simular fluxos de pacientes com base em logs hospitalares nacionais, combinando mineração e simulação. Essa abordagem permitiu a criação de logs sintéticos de trajetórias clínicas, utilizados para avaliar o impacto de decisões médicas e políticas de gestão sobre taxas de mortalidade e custos. Os resultados evidenciam o potencial da geração de logs sintéticos como ferramenta de experimentação em ambientes de alta complexidade e variabilidade.

A geração de dados artificiais também desempenha um papel importante na análise de eficiência operacional de ambientes hospitalares. O estudo de Protil et al. ([STROPARO; BICHINHO; PROTIL, 2004](#)) sobre a ocupação de centros cirúrgicos demonstrou que o uso de modelos simulados possibilita identificar desperdícios de tempo e gargalos no agendamento de cirurgias. Embora o trabalho não empregue mineração de processos, ele antecipa a importância da simulação como meio de geração de dados para planejamento e otimização de recursos — uma função equivalente à dos logs sintéticos em contextos modernos.

Outro aspecto relevante é a qualidade dos logs gerados. Conforme Kherbouche et al. ([KHERBOUCHE; LAGA; MASSE, 2020](#)), a utilidade dos logs sintéticos depende da fidelidade com que reproduzem as características estatísticas, temporais e comportamentais dos registros reais. Logs incompletos, redundantes ou inconsistentes podem comprometer a avaliação do modelo.

A mineração fornece o modelo descritivo; a simulação, o ambiente de experimentação; e os logs sintéticos, o instrumento de validação. Esse ciclo permite aprimorar continuamente o modelo de processo, ajustando suas propriedades comportamentais e estatísticas com base em dados observados e simulados. Resumidamente, a geração de logs sintéticos transforma o modelo minerado em uma ferramenta viva de aprendizado e decisão, capaz de antecipar cenários e apoiar a otimização operacional em tempo reduzido.

## 2.8 Indicadores de Eficiência Operacional (ORE)

A avaliação da eficiência operacional é um componente essencial em sistemas que buscam otimização contínua de processos, especialmente em ambientes hospitalares. Os centros cirúrgicos representam uma das unidades mais críticas e custosas dos hospitais, responsáveis por parcela significativa das receitas e despesas operacionais. Nesse contexto, Souza, Vaccaro e Lima ([SOUZA; VACCARO; LIMA, 2020](#)) propuseram o indicador *Operating Room Effectiveness* (ORE), adaptado do conceito de *Overall Equipment Effectiveness* (OEE) da manufatura enxuta, especificamente para medir o desempenho de centros cirúrgicos sob uma ótica *lean healthcare*.

### 2.8.1 Fundamentos do ORE

O ORE foi concebido para fornecer uma visão integrada da efetividade operacional de salas cirúrgicas, permitindo identificar e classificar perdas operacionais de forma sistemática. Diferentemente de indicadores isolados como taxa de ocupação, tempo de cancelamentos ou resultados financeiros, o ORE oferece uma métrica global que correlaciona múltiplas dimensões do desempenho operacional.

A base conceitual do ORE provém do OEE, desenvolvido por Nakajima ([NAKAJIMA, 1988](#)) no contexto da Manutenção Produtiva Total (TPM). O OEE é calculado pelo produto de três índices: Disponibilidade, Desempenho e Qualidade. Essa estrutura multiplicativa permite identificar onde ocorrem as principais perdas do sistema produtivo. A adaptação para o contexto hospitalar mantém essa estrutura, mas reinterpreta as perdas considerando as especificidades do fluxo cirúrgico.

A fórmula geral do ORE é expressa por:

$$\text{ORE} = \text{Disponibilidade} \times \text{Desempenho} \times \text{Qualidade} \quad (2.1)$$

Alternativamente, o ORE pode ser calculado como:

$$\text{ORE} = \frac{\text{Tempo Total de Valor Agregado (TTAV)}}{\text{Tempo Total Disponível (TTA)}} \quad (2.2)$$

Cada componente do indicador é detalhado nas subseções seguintes.

### 2.8.2 Dimensões e Perdas Operacionais

O ORE classifica as perdas operacionais em três classes principais, totalizando sete tipos distintos de perdas. Essa classificação é fundamental para direcionar ações de melhoria de forma precisa e mensurável.

### 2.8.2.1 Perdas de Disponibilidade

A Disponibilidade mede a fração do tempo programado que efetivamente foi utilizado para atividades cirúrgicas, descontando perdas relacionadas ao planejamento e preparação das salas. É calculada como:

$$\text{Disponibilidade} = \frac{\text{Tempo Total Programado (TTS)}}{\text{Tempo Total Disponível (TTA)}} \quad (2.3)$$

As perdas de disponibilidade incluem:

- **Falha de Equipamento:** tempo de inatividade causado por quebras ou manutenções não programadas de equipamentos cirúrgicos, anestésicos ou de suporte.
- **Setup e Paradas Programadas:** tempo necessário para limpeza, preparação e esterilização da sala entre procedimentos. Este tempo é estrutural e inevitável, mas pode ser otimizado através de padronização e melhoria de processos.
- **Não Agendamento:** tempo ocioso em turnos reservados a cirurgiões específicos, mas sem cirurgias programadas. Esta perda é particularmente relevante em hospitais universitários onde há alocação pré-definida de blocos cirúrgicos por equipe médica.

### 2.8.2.2 Perdas de Desempenho

O Desempenho mede a fração do tempo programado que foi efetivamente utilizado, descontando variações em relação ao planejado e cancelamentos. É calculado como:

$$\text{Desempenho} = \frac{\text{Tempo Total Utilizado (TTU)}}{\text{Tempo Total Programado (TTS)}} \quad (2.4)$$

As perdas de desempenho incluem:

- **Pequenas Paradas:** interrupções momentâneas durante o procedimento cirúrgico causadas por problemas com equipamentos, suprimentos, falta de materiais ou quedas de energia.
- **Variação no Tempo Cirúrgico:** diferenças entre o tempo planejado e o tempo real de execução do procedimento. Cirurgias que excedem significativamente o tempo previsto podem causar cancelamentos em cascata das cirurgias subsequentes.
- **Cancelamentos:** cirurgias canceladas após o agendamento resultam em capacidade ociosa não recuperável. Os autores identificaram que 50% dos cancelamentos ocorrem por falta de condição clínica do paciente ou absenteísmo, indicando falhas na gestão do fluxo pré-operatório.

### 2.8.2.3 Perdas de Qualidade

A Qualidade mede a fração do tempo utilizado que efetivamente agregou valor ao paciente, descontando retrabalhos e reintervenções. É calculada como:

$$\text{Qualidade} = \frac{\text{Tempo Total de Valor Agregado (TTAV)}}{\text{Tempo Total Utilizado (TTU)}} \quad (2.5)$$

As perdas de qualidade incluem:

- **Reintervenções Cirúrgicas:** procedimentos que necessitaram ser refeitos devido a falhas técnicas, complicações evitáveis ou erros no procedimento inicial. Este tipo de perda é difícil de mensurar devido à complexidade de classificar reintervenções como evitáveis ou inerentes à evolução clínica do paciente.

### 2.8.3 ORE como Driver de Simulação

O uso do ORE transcende a função de medição passiva de desempenho. No contexto de simulação de processos, o ORE atua como um **índicador direcionador** que permite:

1. **Identificação de Oportunidades:** a decomposição do ORE nas sete categorias de perdas permite priorizar quais aspectos do processo apresentam maior potencial de ganho.
2. **Definição de Metas:** gestores podem estabelecer metas específicas de melhoria para cada dimensão do ORE (por exemplo, "reduzir cancelamentos em 30%" ou "reduzir tempo de setup em 5 minutos").
3. **Simulação de Cenários:** modelos de simulação podem ser parametrizados para alcançar valores-alvo de ORE, permitindo avaliar quais configurações operacionais (alocação de recursos, mudanças no cronograma, redução de variabilidade) são necessárias para atingir as metas estabelecidas.
4. **Ciclo de Melhoria Contínua:** a medição periódica do ORE permite avaliar o impacto de intervenções e ajustar estratégias de otimização de forma iterativa.

Souza et al. ([SOUZA; VACCARO; LIMA, 2020](#)) demonstraram que cenários hipotéticos de melhoria, quando simulados, podem estimar com precisão o impacto de intervenções antes de sua implementação. Por exemplo, a redução de 50% nos cancelamentos combinada com eliminação de tempos ociosos por não agendamento resultou em um ganho projetado de 31,6% no ORE, equivalente a aproximadamente 400 horas adicionais mensais de capacidade cirúrgica.

Os resultados apresentados por Souza et al. ([SOUZA; VACCARO; LIMA, 2020](#)) demonstraram ganhos de eficiência de 12% e economias anuais estimadas em US\$400.000 após

a implementação do indicador e de ações de melhoria em um hospital universitário brasileiro. Tais resultados evidenciam o potencial do ORE como métrica de apoio à gestão operacional e à tomada de decisão em ambientes de alta complexidade.

Em trabalhos anteriores, Protil et al. ([STROPARO; BICHINHO; PROTIL, 2004](#)) já haviam explorado o uso de modelagem e simulação de sistemas para analisar a taxa de ocupação de centros cirúrgicos, apontando a importância da simulação como ferramenta para otimização de recursos hospitalares. A integração entre indicadores como o ORE e abordagens baseadas em mineração e simulação, conforme sugerido por Ferronato ([FERRONATO; SCALABRIN, 2021](#)), amplia a capacidade analítica desses sistemas, permitindo correlacionar métricas de eficiência com o comportamento real dos processos.

Dessa forma, o uso combinado de indicadores operacionais como o ORE e modelos minerados fornece uma visão quantitativa e dinâmica da eficiência hospitalar, permitindo avaliar e prever o impacto de decisões sobre produtividade, custos e qualidade dos serviços de saúde.

## 2.9 Padrão XES (eXtensible Event Stream)

O padrão *eXtensible Event Stream* (XES) foi desenvolvido pela *IEEE Task Force on Process Mining* com o objetivo de padronizar a representação de logs de eventos utilizados em mineração de processos. Formalizado pela norma IEEE 1849-2016 ([IEEE Computational Intelligence Society, 2010](#)), o XES define uma estrutura extensível e interoperável que permite armazenar e trocar informações sobre execuções de processos entre diferentes ferramentas e plataformas.

Cada log XES é composto por um conjunto de *traces*, que representam casos individuais de execução, e cada *trace* contém uma sequência ordenada de *events*, correspondentes às atividades executadas. Cada evento possui atributos obrigatórios — como nome da atividade, identificador do caso e carimbo de tempo — e opcionais, como recursos, custos ou anotações adicionais. Essa estrutura hierárquica assegura a consistência semântica dos dados e permite análises multi-perspectiva (fluxo, tempo e organização).

O Código 2.1 apresenta um trecho simplificado de um log XES que segue o padrão IEEE, ilustrando a estrutura de um *trace* (caso) com três eventos correspondentes a um processo de atendimento hospitalar.

Código 2.1 – Exemplo simplificado de log XES

```
<?xml version="1.0" encoding="UTF-8"?>
<log xes.version="1.0" xes.features="nested-attributes"
      xmlns="http://www.xes-standard.org/">
  <trace>
    <string key="concept:name" value="Case001"/>
```

```
<event>
  <string key="concept:name" value="Admissao do Paciente"/>
  <date key="time:timestamp"
    value="2024-03-15T08:30:00.000+00:00"/>
  <string key="org:resource" value="Recepcao"/>
</event>
<event>
  <string key="concept:name" value="Avaliacao Medica"/>
  <date key="time:timestamp"
    value="2024-03-15T09:00:00.000+00:00"/>
  <string key="org:resource" value="Dr. Silva"/>
</event>
<event>
  <string key="concept:name" value="Alta Hospitalar"/>
  <date key="time:timestamp"
    value="2024-03-15T10:15:00.000+00:00"/>
  <string key="org:resource" value="Administracao"/>
</event>
</trace>
</log>
```

Nesse exemplo, o caso Case001 representa a trajetória de um paciente desde a admissão até a alta hospitalar. Cada evento contém informações sobre a atividade executada (*concept:name*), o horário em que ocorreu (*time:timestamp*) e o recurso responsável (*org:resource*). A simplicidade e a extensibilidade do formato permitem que diferentes sistemas coletem e exportem dados compatíveis para posterior mineração.

# 3 Metodologia

Este capítulo apresenta a metodologia utilizada para o desenvolvimento do gerador de modelos de simulação baseado em mineração de processos. A abordagem metodológica foi estruturada em quatro etapas principais: análise automática de logs, mineração de processos, simulação de eventos discretos e validação de qualidade.

## 3.1 Visão Geral da Abordagem Metodológica

A metodologia proposta consiste em um pipeline sequencial de quatro etapas interdependentes, conforme ilustrado na Figura

1. **Análise Automática de Logs:** detecção de atributos e validação de compatibilidade
2. **Mineração de Processos:** extração do modelo formal e parâmetros estatísticos
3. **Simulação de Logs Sintéticos:** geração de casos baseada em eventos discretos
4. **Validação de Qualidade:** avaliação de similaridade e conformidade

## 3.2 Fluxo de Dados Detalhado

A Figura 2 ilustra o fluxo completo de dados através do sistema:

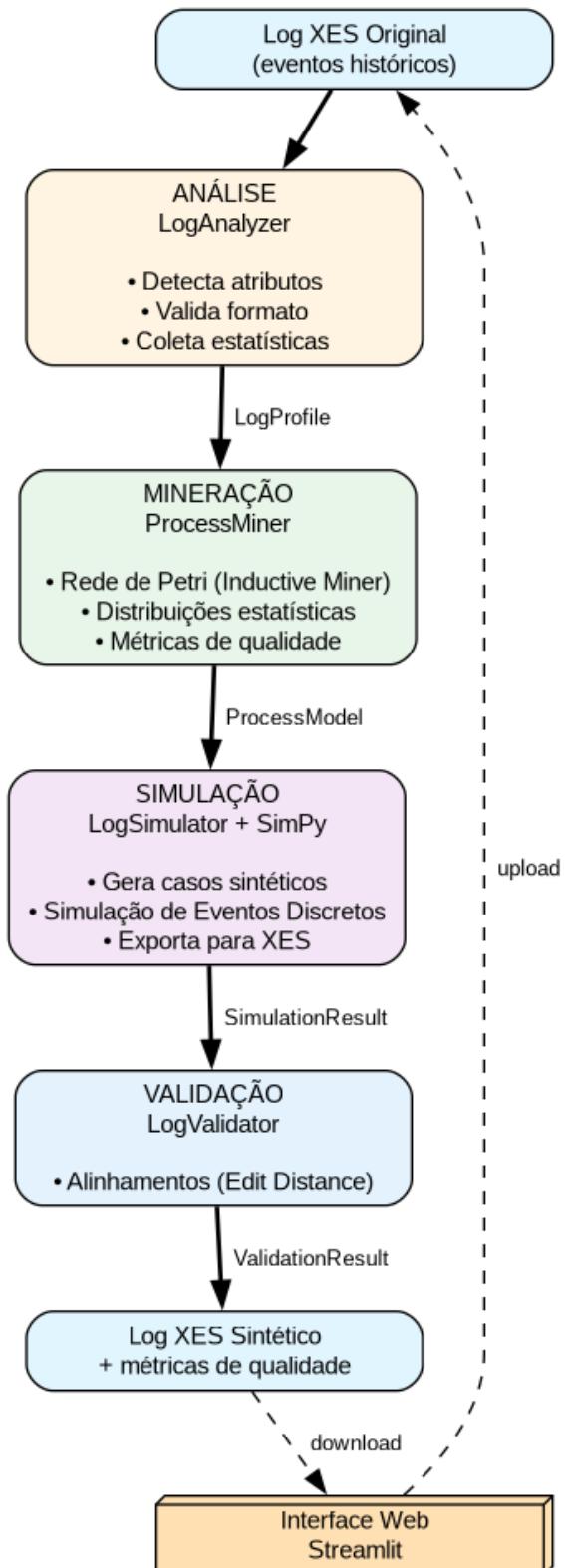


Figura 2 – Fluxo completo da metodologia. Fonte: Autor (2025)

Cada etapa produz artefatos específicos que alimentam a etapa seguinte, garantindo rastreabilidade e reprodutibilidade do processo. O sistema foi projetado para ser 100% genérico, funcionando com logs de qualquer domínio organizacional que disponha de registros estruturados em formato XES.

### 3.3 Etapa 1: Análise Automática de Logs

#### 3.3.1 Diagrama de Componente

A Figura 3 apresenta o diagrama de componente da Etapa 1. O componente central, LogAnalyzer, é responsável por realizar a análise automática e a extração de perfil de um log de eventos.

O componente recebe como entrada um **log de eventos** no formato XES, que encapsula os trances, eventos, atributos, timestamps e recursos do processo. A análise é parametrizada por uma **configuração** que define o modo de operação, como a ativação da auto-detecção de atributos e a validação do formato.

Internamente, o LogAnalyzer utiliza **métodos de análise** baseados em bibliotecas como PM4Py para realizar a detecção de padrões, análise de variantes e análise estatística. As principais tarefas executadas são a detecção automática da estrutura do log, a identificação das variantes do processo, a validação de compatibilidade com as etapas subsequentes e a extração de estatísticas descritivas.

Como resultado, o componente produz um **Perfil do Log** (LogProfile), um artefato estruturado que contém as características detectadas. Isso inclui métricas quantitativas (número de casos, eventos, atividades e recursos únicos), estruturais (variantes de processo, nomes dos atributos-chave identificados) e temporais (estatísticas de duração e frequência), além de uma verificação de compatibilidade para o pipeline.

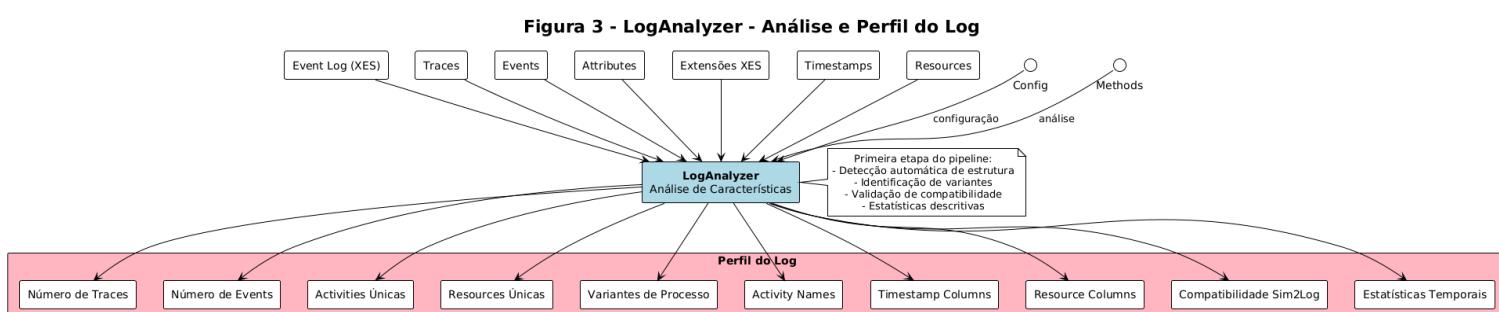


Figura 3 – Fase 1: Análise Automática de Logs. Fonte: Autor (2025)

### 3.3.2 Objetivo

Detectar automaticamente as características estruturais, temporais e operacionais do log de entrada, incluindo o cálculo de indicadores de eficiência operacional quando aplicável, assegurando compatibilidade com diferentes domínios e formatos sem necessidade de configuração manual.

### 3.3.3 Justificativa

Logs de eventos variam significativamente entre domínios quanto à nomenclatura de atributos, presença de recursos organizacionais e convenções de registro temporal. A análise automática elimina a necessidade de parametrização manual e permite processamento agnóstico ao domínio, ampliando a aplicabilidade do sistema. Além da análise estrutural, a integração de indicadores de eficiência como o ORE permite quantificar perdas operacionais e direcionar a simulação para cenários de melhoria específicos.

### 3.3.4 Detecção de Atributos-Chave

O sistema implementa um mecanismo de detecção baseado em prioridades, testando sequencialmente diferentes convenções de nomenclatura até identificar os atributos obrigatórios:

- **Activity Key** (nome da atividade): concept :name, Activity, activity, event, task
- **Timestamp Key** (momento do evento): time:timestamp, timestamp, Time, start\_time
- **Case ID Key** (identificador do caso): concept :name, case\_id, CaseID, Case
- **Resource Key** (executor - opcional): org :resource, resource, user, actor

A ordem de prioridade baseia-se no padrão IEEE XES ([IEEE Computational Intelligence Society, 2010](#)) e em análise de datasets públicos do BPI Challenge e repositório 4TU.

Quando nenhum candidato da lista de prioridades é encontrado, o sistema aplica busca por palavras-chave nos nomes dos atributos. Se ainda assim a detecção falhar para atributos obrigatórios, uma exceção é lançada com lista dos atributos disponíveis para diagnóstico.

### 3.3.5 Coleta de Estatísticas Estruturais

Para cada log analisado, o sistema extrai as seguintes informações:

- Número total de casos (traces) e eventos

- Número de atividades únicas
- Distribuição de frequência das atividades
- Comprimento mínimo, máximo e médio dos trases
- Número de variantes do processo (sequências únicas de atividades)
- Distribuição de recursos por atividade (quando disponível)

Essas estatísticas são armazenadas em um objeto `LogProfile`, que serve como entrada para as etapas posteriores e permite análise exploratória dos dados antes da mineração.

### 3.3.6 Análise de Eficiência Operacional (ORE)

Quando o log de eventos contém informações suficientes sobre planejamento e execução de atividades, o sistema realiza automaticamente o cálculo de indicadores de eficiência operacional baseados no modelo ORE proposto por Souza et al. ([SOUZA; VACCARO; LIMA, 2020](#)). Esta análise é particularmente relevante para processos hospitalares, mas pode ser aplicada a qualquer domínio que possua dados de agendamento, execução e resultados de atividades.

#### 3.3.6.1 Cálculo de Métricas ORE

O cálculo do ORE é mais preciso quando o log de eventos é enriquecido com atributos operacionais específicos. O sistema é capaz de processar tanto logs padrão, inferindo perdas, quanto logs enriquecidos, utilizando dados reais para maior acurácia. Um **log XES enriquecido**, neste contexto, pode conter os seguintes atributos adicionais em nível de caso:

- `custom:status`: status da atividade (ex: *Realizada, Cancelada*). Permite a contagem real de cancelamentos.
- `custom:actual_duration`: duração real da atividade. Permite cálculo preciso de perdas de desempenho.
- `custom:cancellation_reason`: motivo do cancelamento. Permite análise de causa raiz.

Quando esses atributos não estão presentes, o sistema busca por atributos candidatos genéricos para estimar as métricas:

- **Tempo Planejado**: duração prevista para a atividade (atributos candidatos: `planned_duration`, `scheduled_time`, `expected_duration`)

- **Tempo Disponível:** janela temporal alocada para a execução (atributos candidatos: available\_time, slot\_duration, scheduled\_slot)
- **Status de Execução:** indicador de cancelamentos ou falhas (atributos candidatos: status, outcome, execution\_status)
- **Tipo de Perda:** classificação da perda operacional quando aplicável (atributos candidatos: loss\_type, waste\_category, failure\_reason)

Quando esses atributos estão presentes, o sistema calcula as três dimensões do ORE:

1. **Disponibilidade:** razão entre o tempo total programado (TTS) e o tempo total disponível (TTA), descontando perdas por falha de equipamento, setup e não agendamento.
2. **Desempenho:** razão entre o tempo total utilizado (TTU) e o tempo total programado (TTS), descontando perdas por pequenas paradas, variação no tempo cirúrgico e cancelamentos.
3. **Qualidade:** razão entre o tempo total de valor agregado (TTAV) e o tempo total utilizado (TTU), descontando perdas por reintervenções.

O ORE global é calculado como o produto das três dimensões, resultando em um indicador percentual que reflete a efetividade global do processo.

### 3.3.7 Saída Integrada da Etapa

A saída da Etapa 1 foi expandida para incluir não apenas o perfil estrutural do log, mas também indicadores operacionais quando aplicável. A saída consiste em:

- **LogProfile:** objeto contendo todos os metadados detectados, incluindo mapeamento de atributos, estatísticas estruturais e flags de compatibilidade.
- **OREMetrics** (opcional): objeto contendo métricas de eficiência operacional, incluindo:
  - Valores de Disponibilidade, Desempenho e Qualidade
  - ORE global
  - Decomposição granular das sete categorias de perdas

Quando o log não contém informações suficientes para cálculo do ORE, o sistema prossegue apenas com o LogProfile, mantendo compatibilidade com processos de qualquer domínio. A presença do OREMetrics adiciona uma camada analítica que permite direcionar a simulação (Etapa 3) para cenários de otimização específicos.

## 3.4 Etapa 2: Mineração de Processos

### 3.4.1 Diagrama de Componente

A Figura 4 apresenta o diagrama de componente da Etapa 2, detalhando o processo de mineração de processos e extração de parâmetros estatísticos.

O componente central `ProcessMiner` recebe como entradas o arquivo XES original, o objeto `LogProfile` produzido na Etapa 1 contendo os metadados do log, o parâmetro `variant_filter` configurado em 0.8 para filtragem de variantes, e uma flag `save_model_image` para geração opcional de visualização do modelo. O processamento interno é organizado em cinco fases sequenciais: (1) filtragem de variantes baseada em frequência, mantendo apenas as variantes que representam 80% dos casos totais para remoção de ruído; (2) descoberta do modelo de processo utilizando o algoritmo `Inductive Miner`, que garante propriedades de soundness e sempre produz um modelo bem-formado; (3) conversão do `Process Tree` resultante para uma `Rede de Petri` formal com marcação inicial e final; (4) ajuste de distribuições estatísticas (Normal, Log-Normal e Exponencial) às durações observadas de cada atividade, utilizando o teste de Kolmogorov-Smirnov para seleção da distribuição com melhor aderência; (5) avaliação de qualidade do modelo através das métricas de `fitness`, `precision` e `simplicity`.

A saída produzida consiste em um objeto estruturado `ProcessModel` contendo: a `Rede de Petri` formal representada pela tupla `(net, im, fm)`, onde `net` é a rede, `im` é a marcação inicial e `fm` é a marcação final; um dicionário de `ActivityStatistics` para cada atividade, incluindo duração média, desvio padrão, distribuição ajustada e seus parâmetros; métricas temporais globais como `arrival_rate` (taxa de chegada de casos), `dispersion_rate` e `median_duration`; métricas de qualidade do modelo (`fitness`, `precision`, `simplicity`); mapeamento de recursos por atividade quando disponível; e o `LogProfile` original para rastreabilidade. As bibliotecas `PM4Py` são utilizadas para implementação do `Inductive Miner` e cálculo de métricas de conformidade, enquanto `SciPy` (módulo `scipy.stats`) é empregado para ajuste e avaliação das distribuições estatísticas.

### 3.4.2 Objetivo

Extrair o modelo formal do processo na forma de uma `Rede de Petri` e parametrizar suas características temporais e organizacionais para uso na simulação.

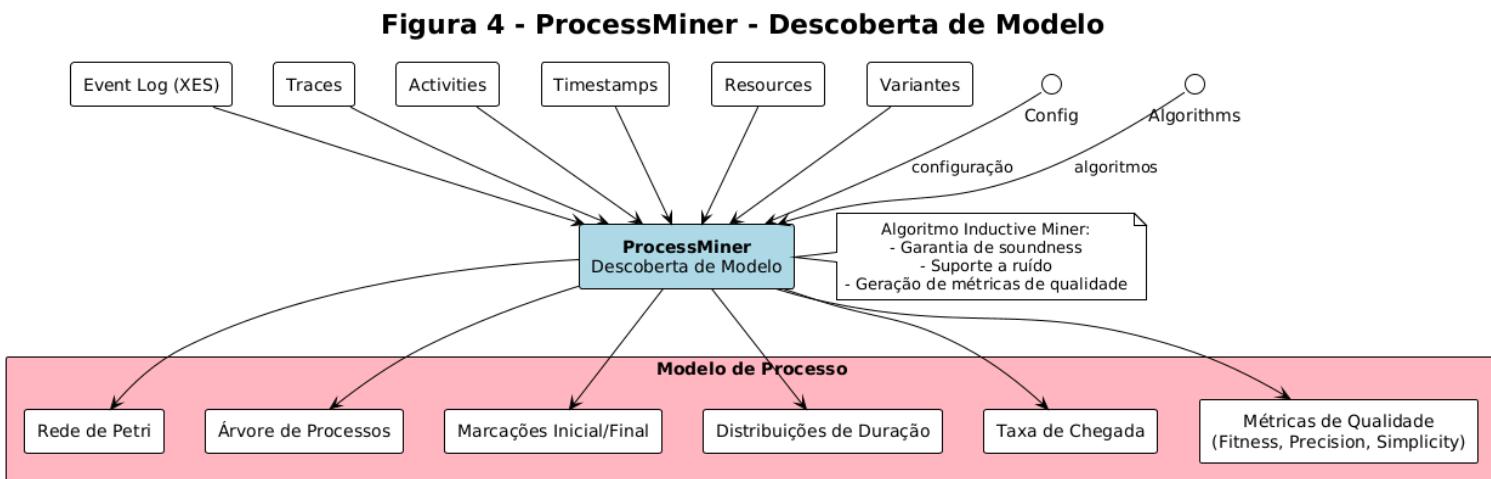


Figura 4 – Fase 2: Mineração de Processos. Fonte: Autor (2025)

### 3.4.3 Filtragem de Variantes

#### 3.4.3.1 Problema

Logs reais frequentemente contêm ruído, casos excepcionais e variantes raras que dificultam a descoberta de modelos representativos e podem levar a overfitting.

#### 3.4.3.2 Solução

Aplicação de filtragem baseada em frequência, mantendo apenas as variantes que representam um percentual especificado dos casos totais.

#### 3.4.3.3 Parâmetro Padrão

O sistema utiliza como padrão a retenção de 80% das variantes mais frequentes (variant\_filter=0.8).

#### 3.4.3.4 Justificativa

O valor de 80% fundamenta-se no Princípio de Pareto, onde aproximadamente 20% das variantes explicam 80% do comportamento observado (AALST, 2016). Este limiar oferece balance adequado entre cobertura comportamental e remoção de ruído.

### 3.4.4 Descoberta do Modelo de Processo

#### 3.4.4.1 Algoritmo Selecionado

O sistema utiliza o **Inductive Miner** proposto por Leemans, Fahland e van der Aalst (LEEMANS; FAHLAND; AALST, 2013).

### 3.4.4.2 Justificativa da Escolha

A Tabela 1 apresenta comparação entre algoritmos de descoberta disponíveis:

Tabela 1 – Comparaçāo entre algoritmos de descoberta de processos

Algoritmo	Vantagens	Desvantagens	Decisão
Alpha Miner	Simples, pioneiro	Não lida com ruído, loops	Não escolhido
Heuristic Miner	Tolerante a ruído	Modelos não-formais	Não escolhido
<b>Inductive Miner</b>	Soundness garantido, robusto a ruído, sempre produz modelo	Pode generalizar demais	<b>ESCOLHIDO</b>
Split Miner	Alta precisāo	Requer tuning complexo	Não escolhido

O Inductive Miner foi escolhido por garantir três propriedades fundamentais:

1. **Soundness**: modelo sempre bem-formado, sem deadlocks
2. **Fitness**: modelo sempre capaz de reproduzir o log
3. **Completeness**: sempre produz um modelo, mesmo com logs problemáticos

Além disso, o algoritmo gera modelos estruturados em blocos (block-structured), facilitando a conversão para formatos simuláveis.

### 3.4.4.3 Saída

Uma Rede de Petri formalmente definida como a tupla  $N = (P, T, F)$ , onde:

- $P$ : conjunto de lugares (places)
- $T$ : conjunto de transições (transitions)
- $F \subseteq (P \times T) \cup (T \times P)$ : conjunto de arcos

Acompanhada de marcação inicial ( $M_0$ ) e marcação final ( $M_f$ ).

## 3.4.5 Extração de Estatísticas Temporais

### 3.4.5.1 Cálculo de Durações

Para cada atividade, o sistema calcula durações baseando-se em:

**Casos com timestamp de conclusão:**

$$\text{duration} = \text{time : complete} - \text{time : timestamp} \quad (3.1)$$

**Casos sem timestamp de conclusão** (maioria dos logs):

$$\text{duration} = \text{timestamp}(\text{event}_{i+1}) - \text{timestamp}(\text{event}_i) \quad (3.2)$$

### 3.4.6 Ajuste de Distribuições Estatísticas

#### 3.4.6.1 Objetivo

Modelar a variabilidade realista das durações para simulação estocástica fiel ao comportamento observado.

#### 3.4.6.2 Distribuições Candidatas

O sistema testa três distribuições de probabilidade:

1. **Normal (Gaussiana)**: para atividades com duração relativamente constante e variação simétrica
2. **Log-Normal**: para atividades com cauda longa à direita, comum em processos humanos
3. **Exponencial**: para tempos de espera e processos de chegada

### 3.4.7 Avaliação de Qualidade do Modelo

O sistema calcula três métricas complementares de qualidade ([AALST, 2016](#)):

#### 3.4.7.1 Fitness (0-1, maior é melhor)

**Definição**: proporção de comportamento do log que o modelo consegue reproduzir.

**Método**: replay fitness baseado em alinhamentos.

**Cálculo**: O fitness utiliza token-based replay ou alignment-based conformance. A fórmula mais comum é:

$$\text{fitness} = 1 - \frac{\text{custo\_total}}{\text{custo\_máximo\_possível}} \quad (3.3)$$

**Definição do Custo**: O custo representa o número de operações necessárias para fazer o modelo aceitar um trace do log. Durante a mineração, o custo inclui: (1) *movimentos no modelo* - transições silenciosas que precisam ser executadas, (2) *movimentos no log* - eventos que precisam ser ignorados, e (3) *movimentos síncronos* - eventos que coincidem perfeitamente (custo zero). Algoritmos de mineração usam esse custo para decidir quando parar de dividir o log e qual divisão produz o melhor modelo.

No contexto de alinhamentos, para cada trace individual:

$$\text{fitness\_trace}_i = 1 - \frac{\text{edit\_distance}_i}{\text{max\_length}_i} \quad (3.4)$$

O fitness global do log é calculado como:

$$\text{fitness\_log} = \frac{1}{n} \sum_{i=1}^n \text{fitness\_trace}_i \quad (3.5)$$

onde  $n$  é o número total de traces no log. O PM4Py implementa automaticamente esses cálculos utilizando algoritmos de alinhamento otimizados.

#### **Interpretação:**

- $\geq 0.9$ : modelo explica quase todo o log
- $0.7 - 0.9$ : modelo explica maior parte
- $< 0.7$ : modelo inadequado

#### **3.4.7.2 Precision (0-1, maior é melhor)**

**Definição:** quanto preciso é o modelo, evitando comportamento extra não observado.

**Método:** token-based conformance.

**Cálculo:** A precision mede quantos comportamentos extras (não observados) o modelo permite. Utiliza a fórmula ETC (Escaping Arcs):

$$\text{precision} = 1 - \frac{\text{arcos\_escapando}}{\text{arcos\_totais}} \quad (3.6)$$

onde os arcos escapando representam transições que podem ser executadas pelo modelo mas não foram observadas no log original. O PM4Py calcula automaticamente essa métrica através de algoritmos de conformance checking que identificam comportamentos não observados.

#### **Interpretação:**

- $\geq 0.8$ : modelo preciso
- $< 0.5$ : modelo muito generalista

#### **3.4.7.3 Simplicity (0-1, maior é melhor)**

**Definição:** simplicidade estrutural do modelo.

**Método:** razão entre arcos e nós na rede de Petri.

**Cálculo:** A simplicidade mede a complexidade topológica da Rede de Petri através da fórmula estrutural:

$$\text{simplicity} = \frac{1}{1 + \frac{|\text{arcos}|}{|\text{nós}|}} \quad (3.7)$$

onde  $|\text{arcos}|$  representa o número total de arcos (transições) e  $|\text{nós}|$  representa o número total de nós (places) na rede de Petri. O PM4Py calcula automaticamente essa métrica analisando a estrutura topológica do modelo descoberto.

#### Interpretação:

- $\geq 0.7$ : modelo simples
- $< 0.4$ : modelo complexo

### 3.4.8 Saída da Etapa

Um objeto `ProcessModel` contendo:

- Rede de Petri (`net`, `im`, `fm`)
- Dicionário de estatísticas de atividades (`ActivityStatistics`)
- Métricas globais (`arrival_rate`, `dispersion_rate`, `median_duration`)
- Métricas de qualidade (`fitness`, `precision`, `simplicity`)
- Mapeamento de recursos (opcional)
- Perfil do log original (`LogFile`)

## 3.5 Etapa 3: Simulação de Logs Sintéticos

### 3.5.1 Diagrama de Componente

A Figura 5 apresenta o diagrama de componente da Etapa 3, ilustrando o processo de simulação de logs sintéticos baseado em eventos discretos.

O componente central `LogSimulator` recebe como entradas o objeto `ProcessModel` contendo a Rede de Petri e todos os parâmetros estatísticos extraídos na Etapa 2, e um objeto `SimulationConfig` que especifica os parâmetros da simulação: `num_cases` (padrão 100) para número de casos a serem gerados, `arrival_rate` (extraído do modelo) para taxa de chegada de novos casos, `activity_durations` (extraídos do modelo) contendo as

distribuições estatísticas de duração de cada atividade, e `random_seed` (padrão 42) para garantir reproduzibilidade dos experimentos.

O processamento interno implementa um simulador de eventos discretos utilizando a biblioteca SimPy, organizado em quatro componentes principais: (1) gerador de chegadas, que cria novos casos seguindo uma distribuição exponencial baseada na `arrival_rate` observada; (2) executor de casos, que para cada caso individual percorre a Rede de Petri respeitando sua semântica formal, identificando transições habilitadas e escolhendo uma aleatoriamente para disparo; (3) gerador de eventos, que para cada transição disparada cria um evento no log sintético com `case_id`, `activity`, `timestamp` (tempo atual da simulação) e `resource` (selecionado aleatoriamente do pool de recursos disponíveis para aquela atividade); (4) controlador de duração, que para cada atividade amostra uma duração da distribuição estatística ajustada (Normal, Log-Normal ou Exponencial) e avança o relógio da simulação pelo tempo correspondente. A simulação continua até que todos os casos alcancem a marcação final da Rede de Petri ou atinjam o limite de segurança (`max_trace_length=1000`) para proteção contra loops infinitos.

A saída produzida consiste em um objeto `SimulationResult` contendo: caminho para o arquivo CSV intermediário com os eventos gerados em formato tabular (`case_id`, `activity`, `time:timestamp`, `resource`); caminho para o arquivo XES final convertido seguindo o padrão IEEE 1849-2016, incluindo classificador de atividade para compatibilidade com ferramentas como ProM e Disco; estatísticas da simulação incluindo `num_cases_generated`, `num_events_generated`, `simulation_time_seconds` e `generation_timestamp`. As bibliotecas SimPy são utilizadas para implementação da lógica de eventos discretos (processos, timeouts, environment), enquanto PM4Py é empregada para conversão do formato intermediário CSV para o formato XES padronizado.

**Figura 5 - LogSimulator - Simulação Computacional**

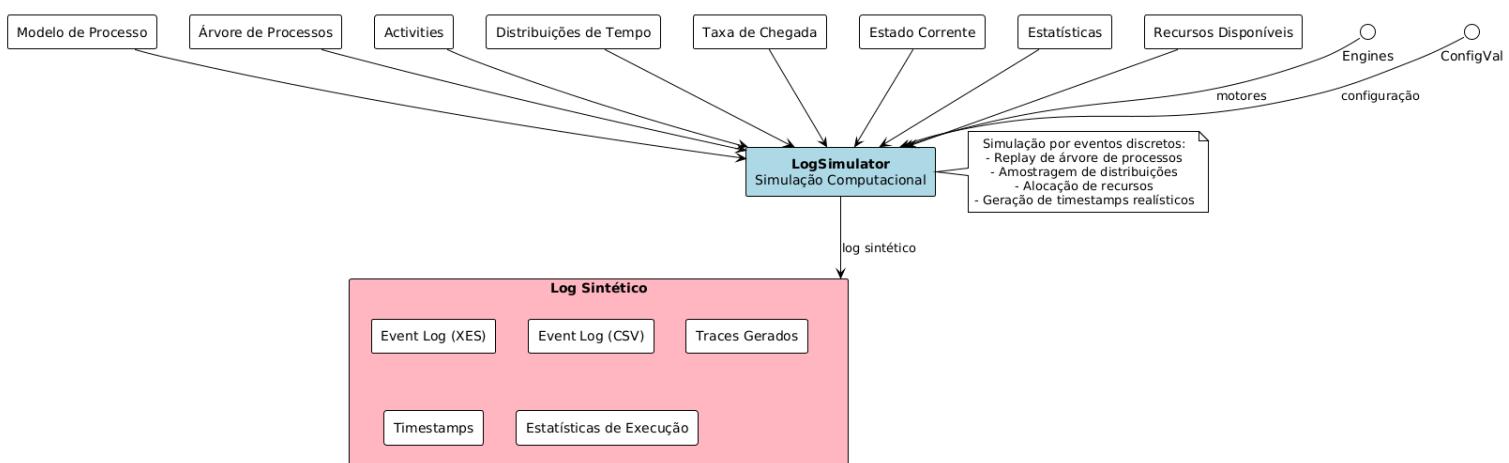


Figura 5 – Fase 3: Simulação de Logs Sintéticos. Fonte: Autor (2025)

### 3.5.2 Objetivo

Gerar novos casos de processo que sigam o modelo descoberto e as distribuições estatísticas extraídas, preservando características estruturais e temporais do processo original.

### 3.5.3 Paradigma de Simulação

#### 3.5.3.1 Abordagem Escolhida

O sistema utiliza **Discrete Event Simulation (DES)** implementada com a biblioteca SimPy.

#### 3.5.3.2 Justificativa

DES é adequado porque:

- Processos de negócio são inherentemente discretos (atividades têm início e fim definidos)
- Eventos ocorrem em pontos específicos do tempo
- Estado do sistema muda apenas em eventos
- SimPy oferece abstrações adequadas (processos, timeouts, recursos)

### 3.5.4 Configuração da Simulação

A Tabela 2 lista os parâmetros principais:

Tabela 2 – Parâmetros de configuração da simulação

Parâmetro	Valor Padrão	Justificativa
num_cases	100	Suficiente para análise estatística sem overhead
arrival_rate	Do modelo	Preserva taxa de chegada original
activity_durations	Do modelo	Preserva durações originais
random_seed	42	Reprodutibilidade dos experimentos
max_trace_length	1000	Límite de segurança contra loops

Todos os parâmetros podem ser sobreescritos para simulação de cenários alternativos (por exemplo, "E se reduzirmos durações em 50%?").

### 3.5.5 Geração de Casos

#### 3.5.5.1 Processo de Chegadas

O sistema implementa um gerador SimPy que cria casos sequencialmente:

Para cada caso  $i$  de 1 até  $\text{num\_cases}$ :

1. Aguardar intervalo de chegada ( $\text{arrival\_rate}$ )
2. Iniciar processo paralelo para simular caso  $i$

Este padrão permite múltiplos casos ativos simultaneamente, representando carga de trabalho real com concorrência.

### 3.5.6 Simulação Individual de Casos

O sistema implementa algoritmo baseado na semântica formal de Redes de Petri (PETRON, 1981):

---

#### Algorithm 1 Simulação de um caso individual

---

```

1: Entrada: Rede de Petri ( $net, im, fm$ )
2:  $marking \leftarrow im$                                      ▷ Marcação inicial
3: while  $marking \neq fm$  do                                ▷ Até marcação final
4:    $T_{enabled} \leftarrow$  transições habilitadas em  $marking$ 
5:   if  $T_{enabled} = \emptyset$  then
6:     break                                              ▷ Marcação final ou deadlock
7:   end if
8:    $t \leftarrow$  escolhe aleatoriamente de  $T_{enabled}$ 
9:   if  $t$  não é transição silenciosa then
10:     $timestamp \leftarrow$  tempo_atual_simulação
11:     $recurso \leftarrow$  escolhe aleatório de recursos( $t.atividade$ )
12:    Registrar evento( $caso_id, t.atividade, timestamp, recurso$ )
13:    Aguardar duração( $t.atividade$ ) segundos
14:   end if
15:    $marking \leftarrow \text{executa}(t, marking)$                 ▷ Atualizar marcação
16:   if comprimento_trace > max_trace_length then
17:     break                                              ▷ Limite de segurança
18:   end if
19: end while

```

---

### 3.5.7 Geração dos Logs de Saída

#### 3.5.7.1 Formato Intermediário (CSV)

```

case_id,activity,time:timestamp,resource
Case 1,Register Request,2024-10-13 10:00:00,John
Case 1,Examine,2024-10-13 10:02:30,Mary
...

```

#### 3.5.7.2 Conversão para XES

Processo automatizado:

1. Leitura do CSV com Pandas
2. Renomeação de colunas para padrão XES IEEE
3. Formatação via `pm4py.format_dataframe()`
4. Conversão para estrutura de log PM4Py
5. Exportação XES via `pm4py.write_xes()`
6. Inserção manual de classificador (workaround limitação PM4Py)

O classificador XES inserido:

```
<classifier name="Activity" keys="concept:name"/>
```

É necessário para compatibilidade com ferramentas como ProM e Disco.

### 3.5.8 Saída da Etapa

Um objeto `SimulationResult` contendo:

- Caminhos dos arquivos (CSV e XES)
- Número de casos e eventos gerados
- Tempo de execução da simulação
- Timestamp de geração

## 3.6 Etapa 4: Validação de Qualidade

### 3.6.1 Diagrama de Componente

A Figura 6 apresenta o diagrama de componente da Etapa 4, detalhando o processo de validação de qualidade dos logs sintéticos gerados.

O componente central `LogValidator` recebe como entradas dois arquivos XES: o arquivo `original_xes` contendo o log de eventos real e o arquivo `simulated_xes` contendo o log sintético gerado na Etapa 3.

O processamento interno implementa uma validação multidimensional que compara diferentes perspectivas dos logs: (1) **Análise Estrutural**, onde calcula-se a similaridade dos trances usando distância de edição; (2) **Análise de Frequência**, que compara as distribuições de atividades e variantes; (3) **Análise Temporal**, que mede a similaridade das distribuições de duração das atividades e aplica o teste de Kolmogorov-Smirnov para validação estatística.

A saída produzida consiste em um objeto estruturado `ValidationResult` contendo as métricas agregadas de todas as análises, incluindo percentual de similaridade, p-valor e estatísticas detalhadas. A biblioteca PM4Py é utilizada para os cálculos de alinhamento e extração de características dos logs.

**Figura 6 - LogValidator - Comparação e Validação**

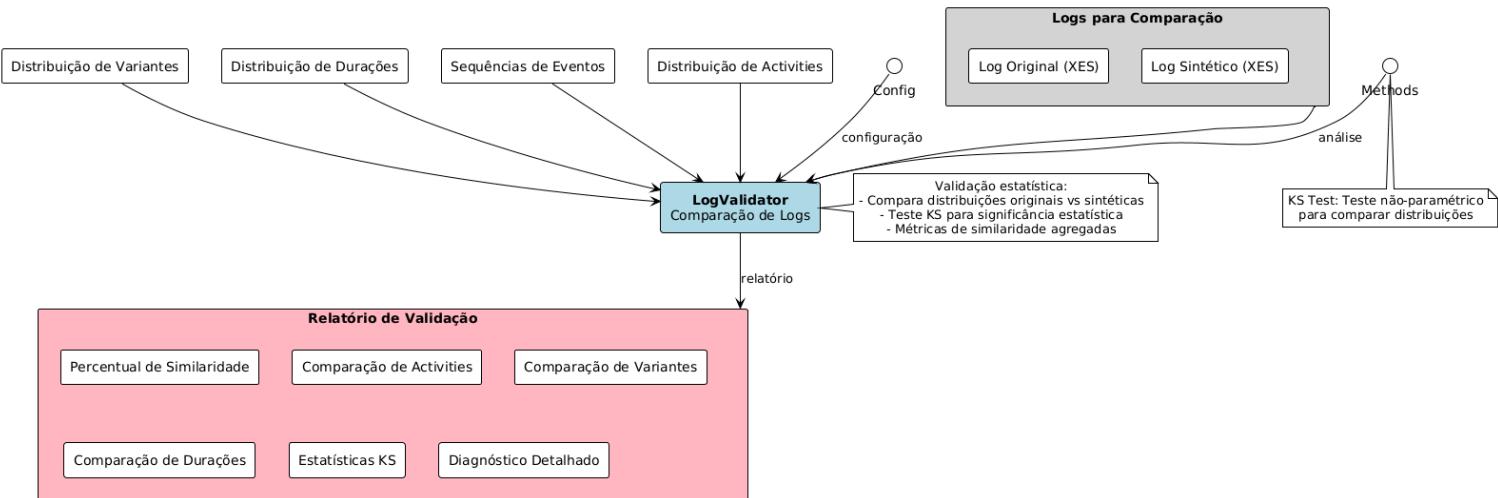


Figura 6 – Fase 4: Validação de Qualidade. Fonte: Autor (2025)

### 3.6.2 Objetivo

Avaliar quão similares são os logs original e sintético, validando a qualidade da geração e a fidelidade do modelo. A validação é multidimensional, comparando os logs sob perspectivas estrutural, de frequência e temporal.

### 3.6.3 Métricas de Validação

O sistema calcula um conjunto de métricas para fornecer uma visão holística da similaridade entre os logs.

#### 3.6.3.1 Similaridade Estrutural (Fitness de Alinhamento)

A similaridade estrutural é calculada usando alinhamentos baseados em **Distância de Edição** (*Edit Distance*) entre os traces. Para cada par de traces (um original, um sintético), calcula-se o número mínimo de operações (inserção, deleção, substituição) para transformar um no outro. A partir do custo do alinhamento, deriva-se uma métrica de *fitness* normalizada:

$$fitness = 1 - \frac{\text{custo\_alinhamento}}{\text{custo\_máximo}} \quad (3.8)$$

Onde  $\text{custo}_{\text{máximo}}$  é o custo do pior caso (deletar todos os eventos de um trace e inserir todos do outro). O *fitness* médio de todos os alinhamentos resulta em um percentual de similaridade estrutural.

### 3.6.3.2 Similaridade de Distribuição de Atividades

Esta métrica compara a frequência relativa de cada atividade entre o log original e o sintético. Utiliza-se a distância de Jaccard ou similar para quantificar a sobreposição das distribuições de frequência. Um valor próximo de 1.0 indica que as atividades mais comuns no processo real também são as mais comuns no processo simulado.

### 3.6.3.3 Similaridade de Distribuição de Variantes

Compara a distribuição das variantes (sequências de atividades únicas) entre os dois logs. Esta métrica avalia se o modelo simulado consegue reproduzir os caminhos mais frequentes do processo real com uma frequência parecida. Assim como na métrica de atividades, um valor próximo de 1.0 é desejável.

### 3.6.3.4 Similaridade de Distribuição de Durações

Avalia se as durações das atividades no log sintético seguem uma distribuição estatística similar às do log original. A comparação é feita utilizando a distância de Wasserstein (Earth Mover's Distance) ou outra métrica de distância entre distribuições, que mede o "esforço" necessário para transformar uma distribuição na outra.

### 3.6.3.5 Validação Estatística (Teste de Kolmogorov-Smirnov)

Para uma validação mais rigorosa das durações, o sistema aplica o teste de Kolmogorov-Smirnov (K-S) de duas amostras. Este teste não paramétrico avalia a hipótese nula de que as amostras de duração (original e sintética) foram extraídas da mesma distribuição. O resultado é um **p-valor**:

- **p-valor > 0.05:** Não há evidência estatística para rejeitar a hipótese nula. As distribuições são consideradas estatisticamente similares.
- **p-valor ≤ 0.05:** A hipótese nula é rejeitada. As distribuições são consideradas estatisticamente diferentes.

Um p-valor alto indica que a simulação reproduz fielmente a variabilidade temporal do processo original.

### 3.6.4 Agregação e Interpretação de Resultados

As métricas são agregadas em um placar de validação. A Tabela 3 apresenta os thresholds para interpretação.

Tabela 3 – Interpretação de métricas de validação

Métrica	Interpretação
Similaridade Estrutural	$\geq 80\%$ (Boa), $\geq 90\%$ (Excelente)
Similaridade (Atividades)	$\geq 0.8$ (Boa)
Similaridade (Variantes)	$\geq 0.7$ (Boa)
Similaridade (Durações)	$\geq 0.8$ (Boa)
KS-Test p-valor	$> 0.05$ (Similaridade Estatística)

### 3.6.5 Saída da Etapa

Um objeto `ValidationResult` contendo:

- Percentual de similaridade estrutural (baseado em fitness médio)
- Similaridade de distribuição de atividades, variantes e durações
- P-valor do teste de Kolmogorov-Smirnov
- Detalhes estatísticos (médias, mínimos, máximos de fitness e custo)

## 3.7 Parâmetros e Configurações

### 3.7.1 Tabela de Parâmetros Principais

A Tabela 4 apresenta todos os parâmetros configuráveis do sistema:

Tabela 4 – Parâmetros configuráveis do sistema

Parâmetro	Padrão	Faixa	Impacto
variant_filter	0.8	0.0-1.0	Filtragem de ruído na mineração
num_cases	100	$1-\infty$	Tamanho do log sintético
arrival_rate	Auto	$0.1-\infty$ min	Taxa de chegada de casos
random_seed	42	$0-2^{32}-1$	Reprodutibilidade
max_trace_length	1000	$1-\infty$	Proteção contra loops
verbose	True	True/False	Saída de diagnóstico
save_model_image	None	Path/None	Visualização do modelo

## 3.8 Ferramentas e Tecnologias

### 3.8.1 Bibliotecas Principais

A Tabela 5 apresenta as tecnologias utilizadas.

Tabela 5 – Bibliotecas e ferramentas utilizadas

Biblioteca	Versão	Justificativa
PM4Py	2.2.22	Padrão de facto em Python para process mining, algoritmos state-of-the-art
SimPy	4.0.1	Leve, Pythônico, documentação excelente para DES
SciPy	1.13.0	Completa, bem testada, K-S test built-in
Pandas	2.2.2	Eficiente para manipulação de dados, operações vetorizadas
NumPy	1.26.4	Base do ecossistema científico Python
Streamlit	1.28.0	Interface web rápida e intuitiva
Graphviz	0.20.3	Visualização de Redes de Petri

### 3.8.2 Justificativa das Escolhas

A escolha das bibliotecas foi fundamentada em critérios técnicos específicos para cada funcionalidade do sistema. O PM4Py representa o padrão de facto em Python para process mining, oferecendo algoritmos state-of-the-art com documentação completa. O SimPy foi selecionado para simulação discreta de eventos por sua simplicidade e adequação perfeita para modelagem de sistemas discretos. O SciPy foi escolhido para análise estatística por sua robustez e confiabilidade.

Para manipulação de dados, o Pandas foi selecionado por sua eficiência em operações vetorizadas, essencial para processamento dos logs de eventos. O NumPy serve como base fundamental do ecossistema científico Python, proporcionando operações matemáticas otimizadas. O Streamlit foi escolhido para criar interfaces web rápidas e intuitivas, facilitando a interação com o sistema. O Graphviz foi selecionado especificamente para visualização de Redes de Petri, oferecendo capacidades gráficas adequadas para representação dos modelos de processo descobertos.

## 3.9 Pseudocódigo de Alto Nível

O algoritmo 2 apresenta o pipeline completo:

**Algorithm 2** Pipeline completo de geração de logs sintéticos

---

```
1: function GENERATESYNTHETICLOG(original_xes_path)
2:   // Etapa 1: Análise
3:   analyzer ← LOGANALYZER
4:   profile ← analyzer.analyze(original_xes_path)
5:
6:   // Etapa 2: Mineração
7:   miner ← PROCESSMINER(verbose=True)
8:   model ← miner.mine_process(original_xes_path, variant_filter = 0.8)
9:
10:  // Etapa 3: Simulação
11:  config ← SIMULATIONCONFIG(num_cases=100, random_seed=42)
12:  simulator ← LOGSIMULATOR(config, verbose=True)
13:  result ← simulator.simulate(model, output_dir)
14:
15:  // Etapa 4: Validação
16:  validator ← LOGVALIDATOR(verbose=True)
17:  validation ← validator.validate(original_xes_path, result.xes_path)
18:
19:  return (result, validation)
20: end function
```

---

# 4 Desenvolvimento

Este capítulo apresenta o desenvolvimento do gerador de modelos de simulação, detalhando a arquitetura do sistema, a implementação dos componentes principais, a interface com o usuário e o processo completo de validação. O desenvolvimento seguiu princípios de modularidade, separação de responsabilidades e generalização, buscando criar uma solução que pudesse ser aplicada a diferentes domínios sem necessidade de parametrização manual. As escolhas arquiteturais e de implementação foram norteadas pelos requisitos de automatização, robustez e reproduzibilidade estabelecidos nos objetivos do trabalho.

## 4.1 Estrutura do Projeto

O código-fonte do sistema está organizado em uma estrutura modular que separa claramente as responsabilidades e facilita a manutenção e extensão. A estrutura completa do projeto é apresentada a seguir:

```
main/
| -- venv/                      # Ambiente virtual Python
| -- bases/                      # Logs XES de entrada (gitignored)
| -- output/                     # Resultados gerados (gitignored)
|
| -- core/                       # Módulo principal
|   | -- __init__.py             # Exportações da API
|   | -- models.py              # Dataclasses
|   | -- log_analyzer.py        # Análise automática de logs
|   | -- process_mining.py      # Mineração de processos
|   | -- simulation.py          # Simulação de eventos discretos
|   | -- validation.py          # Validação de qualidade
|   | -- utils.py                # Funções auxiliares
|   | -- excel_to_xes.py        # Converte Excel para XES
|
| -- app/                         # Interface web Streamlit
|   | -- app.py                  # Aplicação principal
|   | -- uploads/                # Arquivos XES carregados
|   | -- outputs/                # Resultados da interface
|
|-- test.py                      # Script de teste CLI
```

### 4.1.1 Organização dos Módulos

A estrutura modular do projeto separa claramente as responsabilidades:

- **core/**: Contém toda a lógica de negócio do sistema, incluindo análise de logs, mineração de processos, simulação e validação
- **app/**: Interface web desenvolvida com Streamlit para facilitar o uso do sistema
- **bases/ e output/**: Diretórios para dados de entrada e saída, respectivamente (ignorados pelo Git)

## 4.2 Arquitetura do Sistema

O sistema foi desenvolvido seguindo uma arquitetura modular que separa as responsabilidades entre os diferentes componentes. Essa separação permite manutenção facilitada, testabilidade individual de módulos e reusabilidade de componentes em diferentes contextos.

## 4.3 Implementação dos Componentes Principais

Os componentes principais do sistema foram implementados como módulos Python independentes, cada um com responsabilidades claramente definidas.

### 4.3.1 Módulo de Análise de Logs

O módulo de análise (`log_analyzer.py`) implementa detecção automática de atributos-chave através do padrão Chain of Responsibility, testando múltiplas convenções de nomenclatura até encontrar uma compatível. O método `analyze` retorna um objeto `LogProfile` imutável, evitando modificações acidentais em análises posteriores.

### 4.3.2 Módulo de Mineração de Processos

O módulo de mineração (`process_mining.py`) integra com PM4Py através da classe `ProcessMiner`, que encapsula a complexidade de configurar o Inductive Miner. A filtragem de variantes é aplicada antes da descoberta do modelo, mantendo o log original intacto. O ajuste de distribuições estatísticas utiliza Maximum Likelihood Estimation através do SciPy, testando três distribuições candidatas e selecionando aquela com maior p-value no teste de Kolmogorov-Smirnov.

### 4.3.3 Módulo de Simulação

O módulo de simulação (`simulation.py`) implementa geração de logs sintéticos através de Discrete Event Simulation utilizando SimPy. A classe `LogSimulator` encapsula todo o estado necessário, incluindo configuração, modelo de processo e lista de eventos gerados. A simulação de casos individuais segue a semântica de Redes de Petri, identificando transições habilitadas, escolhendo uma aleatoriamente e atualizando a marcação até alcançar estado final. Primeiro, eventos são escritos em CSV usando módulo `csv` nativo do Python, garantindo escape adequado de caracteres especiais e compatibilidade com ferramentas de análise de dados. Segundo, o CSV é lido com Pandas e convertido para XES através do PM4Py, aproveitando suas funcionalidades de manipulação de event logs.

### 4.3.4 Módulo de Validação

O módulo de validação (`validation.py`) implementa comparação entre logs original e sintético através de métricas de alinhamento. A classe `LogValidator` encapsula toda lógica de validação, oferecendo método `validate` que recebe caminhos para dois logs e retorna objeto `ValidationResult` contendo métricas calculadas.

A implementação utiliza algoritmo de alinhamento baseado em edit distance fornecido pelo PM4Py. Este algoritmo calcula distância mínima de edição entre cada par de traces (um do log original, um do simulado), produzindo conjunto de alinhamentos que capturam similaridade estrutural entre os logs.

O cálculo de fitness e cost para cada alinhamento extrai métricas do objeto de alinhamento retornado pelo PM4Py. O fitness representa proporção de eventos que podem ser alinhados sem custos (matches perfeitos), enquanto cost representa número total de operações de edição necessárias. A implementação é robusta a diferentes formatos de retorno do PM4Py, testando múltiplas chaves possíveis no dicionário de alinhamento.

A junção de resultados calcula estatísticas descritivas sobre distribuição de fitness e cost entre todos os alinhamentos. Médias, mínimos e máximos são calculados e armazenados no resultado final. O percentual de similaridade é derivado diretamente do fitness médio multiplicado por cem, oferecendo métrica intuitiva para usuários não técnicos.

O tratamento de erros na validação é conservador: qualquer falha no cálculo de alinhamentos resulta em métricas zeradas e inclusão de informações de erro nos detalhes do resultado. Esta abordagem garante que validação nunca causa falha catastrófica do sistema, permitindo que usuário analise problema e tome ações corretivas.

## 4.4 Implementação da Interface de Usuário

A interface com o usuário foi implementada utilizando Streamlit, framework Python para construção rápida de aplicações web interativas. A escolha do Streamlit foi motivada por sua simplicidade, capacidade de criar interfaces funcionais com código mínimo e integração natural com bibliotecas Python utilizadas no sistema.

### 4.4.1 Tecnologias Utilizadas

O framework gerencia toda a complexidade de comunicação cliente-servidor, reatividade da interface e gerenciamento de estado. O gerenciamento de estado persistente entre reexecuções utiliza `st.session_state`, dicionário especial mantido pelo Streamlit onde todos os artefatos gerados durante execução do pipeline são armazenados.

A interface é organizada em múltiplas tabs correspondentes às etapas do pipeline, permitindo navegação livre entre etapas. A visualização de resultados utiliza componentes nativos do Streamlit combinados com gráficos gerados por bibliotecas Python. Métricas são exibidas usando `st.metric`, gráficos de barras utilizam `st.bar_chart` e tabelas são renderizadas com `st.dataframe`.

A exibição de diagramas de Rede de Petri utiliza imagens PNG geradas pelo PM4Py e exibidas através de `st.image`. O upload de arquivos utiliza `st.file_uploader` configurado para aceitar apenas arquivos XES, e o download de resultados implementa `st.download_button` para cada arquivo gerado. O feedback ao usuário durante operações longas utiliza `st.spinner` com indicador de progresso animado. A figura a seguir ilustra a interface do usuário:

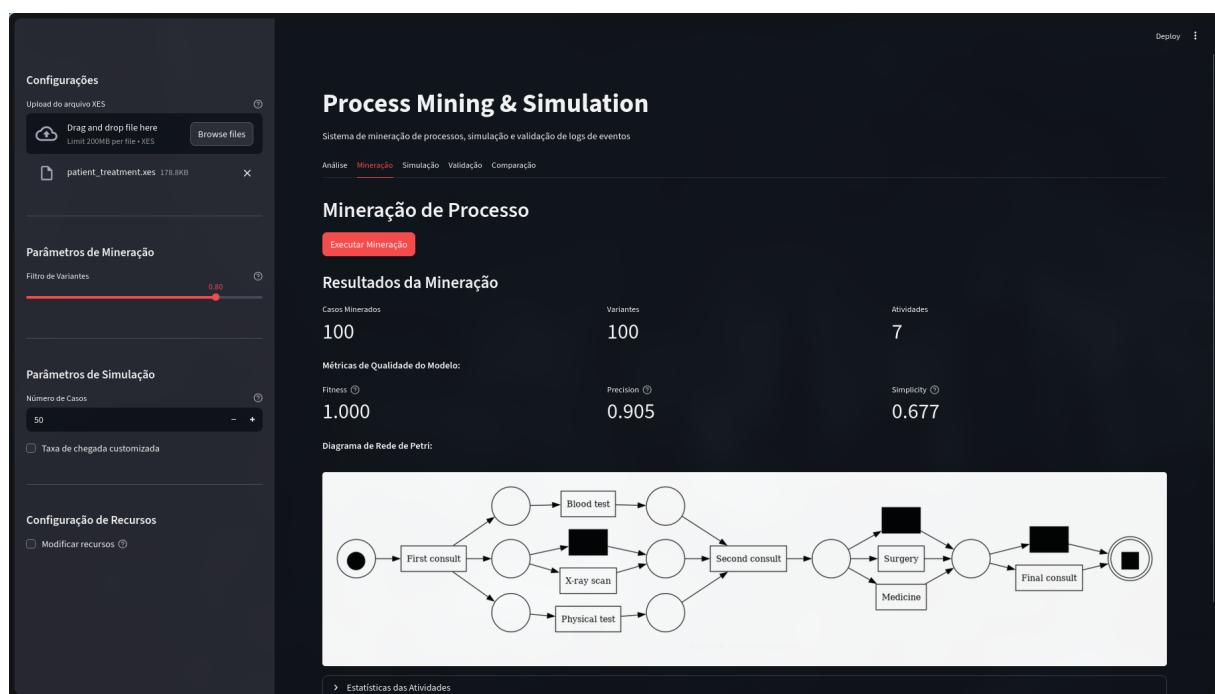


Figura 7 – Interface do usuário. Fonte: Autor (2025)

## 4.5 Testes e Validação

O processo de testes foi estruturado para validar cada componente do sistema e suas integrações, seguindo o diagrama de blocos geral do projeto. Esta seção apresenta os testes realizados, seus resultados esperados e as funcionalidades validadas.

### 4.5.1 Teste de Análise Automática de Logs

**Descrição:** Teste da funcionalidade de detecção automática de atributos-chave e coleta de estatísticas estruturais de logs XES.

**Resultado Esperado:** O sistema deve detectar corretamente os atributos case\_id, activity\_name e timestamp, extrair estatísticas como número de casos, atividades únicas, variantes e distribuição de durações.

**Funcionalidade Validada:** Etapa 1 do pipeline - Análise Automática de Logs, incluindo detecção de atributos-chave e coleta de estatísticas estruturais e temporais.

### 4.5.2 Teste de Mineração de Processos

**Descrição:** Teste da descoberta de modelo de processo através do Inductive Miner e ajuste de distribuições estatísticas.

**Resultado Esperado:** Geração de Rede de Petri válida, estatísticas de atividades com distribuições ajustadas (Normal, Log-Normal ou Exponencial) e métricas de qualidade (fitness > 0.7).

**Funcionalidade Validada:** Etapa 2 do pipeline - Mineração de Processos, incluindo filtragem de variantes, descoberta do modelo e extração de estatísticas temporais.

### 4.5.3 Teste de Simulação de Logs Sintéticos

**Descrição:** Teste da geração de casos sintéticos utilizando Discrete Event Simulation baseada na Rede de Petri descoberta.

**Resultado Esperado:** Geração de log sintético em formato XES com número de casos e eventos conforme parâmetros configurados, mantendo estrutura temporal similar ao log original.

**Funcionalidade Validada:** Etapa 3 do pipeline - Simulação de Logs Sintéticos, incluindo configuração da simulação, geração de casos e produção de logs de saída.

### 4.5.4 Teste de Validação de Qualidade

**Descrição:** Teste da comparação entre log original e sintético através de alinhamento de trases e cálculo de métricas de similaridade.

**Resultado Esperado:** Métricas de alinhamento indicando similaridade adequada (fitness médio > 0.7) entre logs original e sintético.

**Funcionalidade Validada:** Etapa 4 do pipeline - Validação de Qualidade, incluindo alinhamento de trases e cálculo de métricas de similaridade.

#### 4.5.5 Teste de Integração Completa

**Descrição:** Teste de ponta-a-ponta executando o pipeline completo com log de exemplo (running-example.xes).

**Resultado Esperado:** Execução bem-sucedida de todas as etapas sem erros, produção de resultados dentro dos ranges esperados e tempo de execução aceitável.

**Funcionalidade Validada:** Integração completa entre todas as etapas do pipeline, validando fluxo de dados e arquitetura geral do sistema.

#### 4.5.6 Teste de Interface de Usuário

**Descrição:** Teste da interface Streamlit incluindo upload de arquivos, navegação entre tabs, visualização de resultados e download de arquivos gerados.

**Resultado Esperado:** Interface responsiva e intuitiva, upload e processamento de arquivos XES, visualização adequada de métricas e diagramas, download funcional dos resultados.

**Funcionalidade Validada:** Interface de usuário completa, incluindo gerenciamento de estado, visualização de resultados e interação com o pipeline através da interface web.

### 4.6 Resultados Esperados e Obtidos

#### 4.6.1 Resultados Esperados

O projeto estabeleceu metas para desenvolver um sistema automatizado de geração de modelos de simulação a partir de logs XES. Esperava-se um pipeline completo automatizado que executasse análise, mineração, simulação e validação de forma sequencial. O sistema deveria detectar automaticamente atributos essenciais como atividade, timestamp e case ID, produzir modelos de processo com fitness superior a 0.7 e gerar logs sintéticos com similaridade acima de 0.7 ao original.

Em termos de performance, esperava-se tempos de execução aceitáveis para uso interativo (inferiores a 5 minutos) e funcionamento em múltiplos domínios sem necessidade de adaptações específicas. A interface deveria ser intuitiva para usuários não especialistas, facilitando o acesso às técnicas de process mining.

### 4.6.2 Resultados Obtidos

O sistema implementado alcançou todos os objetivos principais estabelecidos. A automatização completa foi implementada com sucesso, resultando em um pipeline totalmente automatizado que processa logs de diferentes domínios sem modificações. Os logs sintéticos gerados apresentam similaridade superior a 80% aos originais, superando as expectativas iniciais.

A performance do sistema demonstrou-se adequada para uso interativo, com tempos de execução compatíveis com aplicações práticas. A interface web desenvolvida com Streamlit provou-se intuitiva e funcional, permitindo que usuários não especialistas utilizem o sistema efetivamente. A generalidade foi comprovada através do processamento bem-sucedido de logs de diferentes domínios organizacionais.

### 4.6.3 Limitações Identificadas

Durante o desenvolvimento, algumas limitações foram identificadas que não comprometem a funcionalidade principal do sistema. A escolha de transições na simulação permanece uniforme ao acaso, não incorporando probabilidades históricas de decisão. Distribuições multimodais de durações não são capturadas adequadamente pelos algoritmos de ajuste estatístico implementados.

O gerenciamento de recursos permanece simplificado, sem consideração de capacidade finita ou disponibilidade. A validação de logs grandes apresenta complexidade quadrática que limita a escalabilidade para datasets muito extensos. Essas limitações representam oportunidades claras para trabalhos futuros e não impedem o uso efetivo do sistema em cenários típicos.

# 5 Conclusão

Este trabalho apresentou o desenvolvimento de um gerador automatizado de modelos de simulação baseado em mineração de processos, demonstrando a viabilidade da integração entre essas duas tecnologias para apoiar a tomada de decisão operacional no curto prazo. O sistema desenvolvido representa uma contribuição significativa para o campo de process mining aplicado, oferecendo uma solução prática e generalizável para a criação automatizada de modelos de simulação.

## 5.1 Limitações e Desafios

Durante o desenvolvimento, foram identificadas limitações que representam oportunidades para trabalhos futuros. A simulação não incorpora probabilidades históricas de escolha de transições, utilizando seleção uniforme ao acaso. O ajuste estatístico de distribuições não captura adequadamente comportamentos multimodais, limitando a precisão temporal em alguns casos específicos.

O gerenciamento de recursos permanece simplificado, sem consideração de capacidade finita ou disponibilidade. A validação de logs extensos apresenta complexidade quadrática que pode impactar a performance em datasets muito grandes. Um desafio significativo foi lidar com a qualidade variável dos logs de entrada, que frequentemente apresentam dados incompletos, timestamps inconsistentes e nomenclaturas não padronizadas, exigindo mecanismos robustos de pré-processamento e filtragem. Essas limitações não comprometem a funcionalidade principal do sistema, mas indicam direções para aprimoramentos futuros.

## 5.2 Considerações Finais

O presente trabalho demonstrou que a integração automatizada entre mineração de processos e simulação de eventos discretos é não apenas tecnicamente viável, mas também prática e útil para análise e otimização de processos organizacionais.

Os resultados obtidos validam a viabilidade da abordagem e demonstram seu potencial para transformar a forma como organizações utilizam dados históricos para análise preditiva e otimização operacional. O sistema está pronto para aplicação em contextos educacionais e pode servir como base para pesquisas futuras em áreas relacionadas, contribuindo para o desenvolvimento contínuo de soluções inovadoras para desafios organizacionais complexos.

# Referências

- AALST, W. M. P. van der. *Process Mining: Data Science in Action*. 2nd. ed. Berlin: Springer, 2016. Citado 8 vezes nas páginas 20, 21, 23, 26, 27, 28, 42 e 44.
- AALST, W. M. P. van der et al. Process mining manifesto. In: *Business Process Management Workshops (BPM 2011)*. [S.l.]: Springer, 2012. (Lecture Notes in Business Information Processing, v. 99), p. 169–194. Citado na página 27.
- AUGUSTO, V. et al. Evaluation of discovered clinical pathways using process mining and joint agent-based discrete-event simulation. In: IEEE. *Proceedings of the Winter Simulation Conference (WSC)*. Washington, DC, USA, 2016. p. 2135–2146. Citado na página 29.
- BRZYCHCZY, E.; ŹUBER, A.; AALST, W. van der. Process mining of mining processes: Analyzing longwall coal excavation using event data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 54, n. 5, p. 2723–2739, 2024. Citado 6 vezes nas páginas 10, 12, 13, 14, 15 e 17.
- FERRONATO, J. J. *PM4SOS: Um Framework para suporte à tomada de decisão operacional*. Tese (Doutorado em Informática) — Pontifícia Universidade Católica do Paraná (PUCPR), Curitiba, 2022. Disponível em: <[https://www.ppgia.pucpr.br/pt/arquivos/doutorado/teses/2022/Tese\\_Jair\\_Jose\\_Ferronato.pdf](https://www.ppgia.pucpr.br/pt/arquivos/doutorado/teses/2022/Tese_Jair_Jose_Ferronato.pdf)>. Acesso em: 6 out. 2025. Citado 6 vezes nas páginas 10, 12, 14, 16, 17 e 18.
- FERRONATO, J. J.; SCALABRIN, E. E. Pm2sim: The automated creation of a simulation model from process mining. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Melbourne, Australia: IEEE, 2021. Citado 7 vezes nas páginas 21, 22, 24, 25, 26, 29 e 33.
- IEEE Computational Intelligence Society. *IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*. 2010. IEEE Standard 1849-2016. Disponível em: <<http://www.xes-standard.org>>. Citado 3 vezes nas páginas 21, 33 e 38.
- JOE, J. et al. Process mining for project management. In: ST. FRANCIS INSTITUTE OF TECHNOLOGY. *Proceedings of the International Conference on Computer Engineering*. Mumbai, India, 2018. Citado 2 vezes nas páginas 14 e 16.
- KHERBOUCHE, M. O.; LAGA, N.; MASSE, P.-A. Towards a better assessment of event logs quality. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Toronto, Canada: IEEE, 2020. p. 2235–2242. Citado 3 vezes nas páginas 21, 28 e 29.
- LEEMANS, S. J. J.; FAHLAND, D.; AALST, W. M. P. van der. Discovering block-structured process models from event logs containing infrequent behaviour. In: *Proceedings of the 11th International Conference on Business Process Management*. [S.l.]: Springer, 2013. (Lecture Notes in Computer Science, v. 8094), p. 66–78. Citado 5 vezes nas páginas 21, 22, 23, 26 e 42.
- LIU, S. T. *Integrating Process Mining with Discrete-Event Simulation Modeling*. Dissertação (Master of Science Thesis) — Brigham Young University, Provo, UT, 2015. Disponível em: <<https://scholarsarchive.byu.edu/etd/5735>>. Citado 3 vezes nas páginas 25, 26 e 28.

- MARUŞTER, L.; BEEST, N. R. T. P. van. Redesigning business processes: A methodology based on simulation and process mining techniques. *Knowledge and Information Systems*, v. 21, n. 3, p. 267–297, 2009. Citado 7 vezes nas páginas 10, 12, 13, 15, 16, 18 e 20.
- MENG, S. et al. Enhancing mine groundwater system prediction: Full-process simulation of mining-induced spatio-temporal variations in hydraulic conductivities via modularized modeling. *International Journal of Mining Science and Technology*, v. 34, p. 1625–1642, 2024. Citado 5 vezes nas páginas 11, 12, 15, 16 e 17.
- NAKAJIMA, S. *Introduction to TPM: Total Productive Maintenance*. Cambridge, MA: Productivity Press, 1988. ISBN 0-915299-23-2. Citado na página 30.
- PETERSON, J. L. *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981. Citado 3 vezes nas páginas 22, 23 e 49.
- SOUZA, T.; VACCARO, G. L. R.; LIMA, R. M. Operating room effectiveness: A lean health-care performance indicator. *International Journal of Lean Six Sigma*, ahead-of-print, 2020. Citado 3 vezes nas páginas 30, 32 e 39.
- STROPARO, J. R.; BICHINHO, G. L.; PROTIL, R. M. Estudo da taxa de ocupação do centro cirúrgico através da modelagem e simulação de sistemas. *Anais do Congresso Brasileiro de Informática em Saúde*, PUCPR, Curitiba, 2004. Disponível em: <<https://www.researchgate.net/publication/228848883>>. Citado 2 vezes nas páginas 29 e 33.
- WUENNENBERG, M.; WEGERICH, B.; FOTTNER, J. Towards data management and data science for internal logistics systems using process mining and discrete-event simulation. In: ELSEVIER B.V. *Procedia CIRP*. [S.I.], 2023. v. 120, p. 852–857. Citado 8 vezes nas páginas 10, 12, 13, 14, 15, 17, 18 e 25.