PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

ESCOLA POLITÉCNICA

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA (PPGIa)

PDG: *PROCESS DRIFT GENERATOR*

CAIO RADUY

CURITIBA
2024

CAIO RADUY

PDG: *PROCESS DRIFT GENERATOR*

Projeto apresentado ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção o título de Mestre em Informática Aplicada, sob orientação do Prof.
Prof. **Dr. Edson Emílio Scalabrin**

CURITIBA
2024

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

R132p
2024

Raduy, Caio
PDG : *process drift generator* / Caio Raduy ; orientador: Edson Emílio Scalabrin. – 2024.
68 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2024
Bibliografia: f. 138-144

1. Informática. 2. Mineração de dados (Computação). 3. Mineração de processos. 4. Processamento eletrônico de dados. I. Scalabrin, Edson Emílio. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD. 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 03 de maio de 2024.

34-2024

DECLARAÇÃO

Declaro para os devidos fins, que **Caio Raduy** defendeu a dissertação de Mestrado intitulada "**Process Drift Generator"**, na área de concentração Ciência da Computação no dia 11 de abril de 2024, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

gov.br
Documento assinado digitalmente
EMERSON CABRERA PARAISO
Data: 08/05/2024 13:44:17-0300
Verifique em https://validar.iti.gov.br

_____
Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

AGRADECIMENTOS

A vida é uma oportunidade. Quero expressar minha gratidão a Deus por me dar a oportunidade de viver todos os dias e por me guiar na direção certa, colocando as pessoas certas na minha vida. A minha família, que sempre acreditou em mim, merece um agradecimento especial. Aos meus pais que sempre me apoiaram. A minha mãe, que está presente diariamente no meu dia a dia, me motivando com seu astral maravilhoso. Ao meu pai por me passar o exemplo de honestidade, seriedade e por me apoiar. A minha irmã, com seu humor contagiante, alegria, risadas e suporte. Sou imensamente grato pela oportunidade que Deus me deu para chegar até aqui, conhecer essas pessoas, nesta universidade, nesta cidade. Agradecer ao processo que tive que passar para entender que agora estou pronto para enfrentar o que o Senhor reservou para mim.

Gostaria de agradecer ao Dr. Edson e à Dra. Denise por acreditarem em mim, pela oportunidade que me deram, pelo tempo que se dedicaram e por me auxiliarem. Ao Dr. Edson por me auxiliar em inúmeros códigos fontes e sempre se mostrar disponível com um astral contagiante. À Dra. Denise por me aguentar [...] com inúmeras mensagens de dúvidas em códigos, e artigos e, sempre, me auxiliar da melhor maneira possível. A Deus que me dá saúde para buscar meus objetivos e sempre me coloca no lugar certo com as pessoas certas. A Deus por conhecer essas duas pessoas maravilhosas que mudaram minha vida. À minha família por acreditar em mim e me apoiar. Ao PIBIC Master e ao PPGIa que deram a oportunidade da minha vida.

Dedico esse trabalho a Deus que me dá a oportunidade diária de viver. Aos meus pais por acreditarem em mim, a minha irmã por sempre me apoiar. Ao Dr. Edson Emílio Scalabrin, a quem tive a oportunidade de conhecer e conviver, um exemplo de pessoa, professor, amigo, mentor e orientador. A Dra. Denise Maria Vecino Sato que sempre me apoiou e recebeu um estudante de Engenharia Civil como orientado em 2020 e desde lá sempre me apoiou.

RESUMO

Introdução: Os processos de negócios são sequências de atividades organizadas para atingir objetivos específicos dentro de uma estrutura empresarial, registradas em logs de eventos por sistemas de informação. Contudo, devido à dinâmica inerente aos processos, é comum que ocorram mudanças na sequência de atividades ao longo do tempo, conhecidas como "*process drifts*". Detectar e compreender essas mudanças é crucial para a gestão eficaz dos processos de negócios. Embora existam métodos para detecção de *drifts*, muitas vezes a validação desses métodos é limitada pela falta de logs de eventos que representem as complexidades dos processos reais. **Objetivo**: O objetivo do projeto é desenvolver um método computacional para gerar conjuntos de bases sintéticas que contenham logs de eventos com *process drifts*. **Revisão Sistemática:** Também foi conduzida uma revisão sistemática da literatura para mapear logs de eventos com *drifts* e geradores de logs existentes. Essa revisão forneceu uma base teórica para o desenvolvimento do método proposto e não encontrou ferramentas que possibilitam a fácil geração de logs de eventos com *drifts* próximos a logs reais. **Método**: Foram desenvolvidas diferentes estratégias, utilizando o ambiente PM4PY, para gerar logs de eventos. O **PDG** é capaz de produzir logs com quatro tipos principais de *drifts*: abruptos, graduais, repentinos e recorrentes, a partir do modelo do processo, da localização e do tipo de *drifts* informados, além do tamanho desejado do log de eventos. Também foi possível configurar a taxa de ruído nos logs e para drifts graduais, a função de decaimento linear ou exponencial. **Logs de eventos gerados pelo PDG**: A validação dos logs foi dividida em duas etapas: inicialmente, foram geradas 10 réplicas para cada log de uma base de dados existente, totalizando 170 logs de eventos com *drifts*. Essas réplicas foram comparadas com os logs originais utilizando o teste de *Wilcoxon*. Em seguida, foi gerado também um conjunto de logs sintéticos com diferentes parâmetros, validando-os posteriormente com a análise das transições abruptas, graduais e investigações dos logs. **Comparação de Ferramentas de Detecção de Drifts**: Foi aplicado o protocolo de testes bem definido a seis abordagens de detecção de *drifts* (VDD *system*, Apromore-*ProDrift fixed*, Apromore- *ProDrift adaptive*, ProM-*Concept Drift*, *Adaptive* IPDD *trace by trace*, IPDD *fixed*.), utilizando dois conjuntos de logs de eventos existentes na literatura e uma base de dados gerada pelo **PDG**. Os resultados destacaram a sensibilidade das ferramentas de detecção baseadas em janelamento fixo ao tamanho da janela, bem como a influência da escolha da janela inicial nos métodos adaptativos e dos *f-scores* médios pelo conjunto de logs de eventos. **Conclusões**: O **PDG** foi implementado com sucesso, permitindo a geração de logs de eventos com diferentes características, incluindo tamanhos variados, intervalos e tipos de drifts, além de ruído. A implementação de *drifts* graduais com decaimento linear ou exponencial foi possível por meio do estudo das funções de probabilidades. A validação realizada com as réplicas de logs existentes e o novo conjunto de logs mostrou a consistência e eficácia do *Process Drift Generator* (**PDG**). Além disso, a comparação das ferramentas salientou a importância do conjuntos de logs de eventos gerados. **Trabalhos futuros**: Sugere-se a replicação da comparação das ferramentas utilizando o *F-score* combinado com o *mean delay* para comparar a distância dos *drifts* reais aos *drift* detectados.

Palavras-chave: Mineração de Processo, *Process Drift*, *Concept Drift*, Log de Eventos, Geração de Logs de Eventos.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS

ADWIN *Adaptive Windowing*

AIMED *Automatic and Incremental approach for business process Model rEpair under concept Drift*

BLA Base de log de eventos artificias existente

BPIC *Business Process Intelligence Challenge*

CDESF *Concept Drift in Event Stream Framework*

CE Critério de Exclusão

CI Critério de Inclusão

CDLG *Concept Drift Log Generator*

CPN *Coloured Petri Net*

DALG *Data Aware Event Log Generator*

DSRM *Design Science Research Methodology*

ET *Error Tolerance*

E Logs de eventos reais

FN *False Negative*

FP *False Positive*

FWIN *Fixed Windowing*

IPDD *Interactive Process Drift Detection*

MSE *Mean Squared Error*

LCDD *Local Complete-based Drift Detection*

PDG *Process Drift Generator*

PELT *Pruned Exact Linear Time*

PLG *Process Log Generator*

PPs Perguntas de Pesquisa

RD *Real Drift*

RMSLE *Root Mean Squared Logarithmic Error*

RT-PLG *Real Time Process Log Generator*

TP *True Positive*

TPCDD *Tsinghua Process Concept Drift Detection*

VDD *Visual Drift Detection System*

WS *Window Size*

SUMÁRIO

1 INTRODUÇÃO

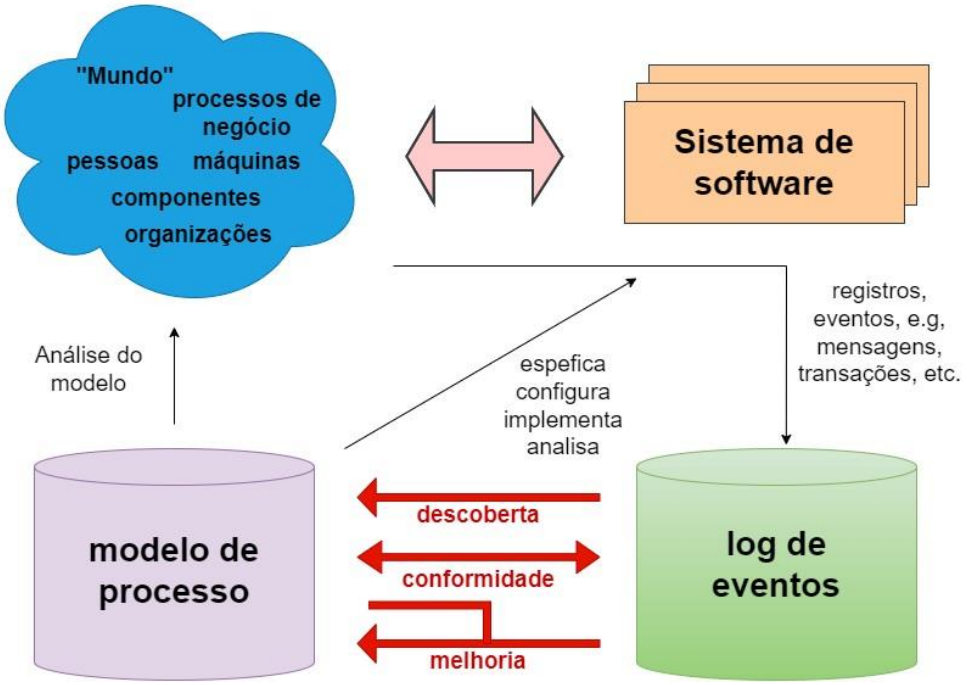
A condução eficiente de um negócio depende da compreensão profunda e da melhoria contínua dos processos que o sustentam. Cada processo empresarial é uma sequência estruturada de atividades com um objetivo final específico, e a execução dessas atividades é registrada nos sistemas de informação da organização. Esses registros, conhecidos como logs de eventos, capturam uma variedade de dados cruciais, como o momento da execução, os responsáveis, as atividades realizadas e o contexto envolvido.

A análise do conteúdo desses logs de eventos é vital para se obter percepções importantes sobre a eficiência e eficácia dos processos empresariais. A mineração de processos surge como uma abordagem inevitável nesse contexto, permitindo extrair conhecimento útil a partir dos registros/logs de eventos. Como destacado em Zheng, Wen e Wang (2017), a mineração de processos é uma ferramenta muito importante para o desenvolvimento empresarial, fornecendo uma visão aprofundada das operações organizacionais.

A mineração de processos permite explorar um log de eventos em três perspectivas diferentes apresentado em Aalst (2016): descoberta, conformidade e aprimoramento. Na perspectiva da descoberta, os dados do log de eventos são explorados para construir modelos de processos sem a necessidade de conhecimento prévio sobre a estrutura do processo. A perspectiva da conformidade visa comparar a execução real do processo com o modelo estabelecido, garantindo que as operações estejam em conformidade com os padrões desejados. Por fim, a perspectiva do aprimoramento utiliza informações do log de eventos para refinamento contínuo dos modelos de processos, impulsionando a melhoria contínua da eficiência e eficácia operacional.

A Figura 1 ilustra visualmente as três perspectivas da mineração de processos e sua interação com os logs de eventos e os sistemas de informação, destacando a importância desses elementos na compreensão e otimização dos processos empresariais.

Figura 1: Três tipos de mineração de processos.



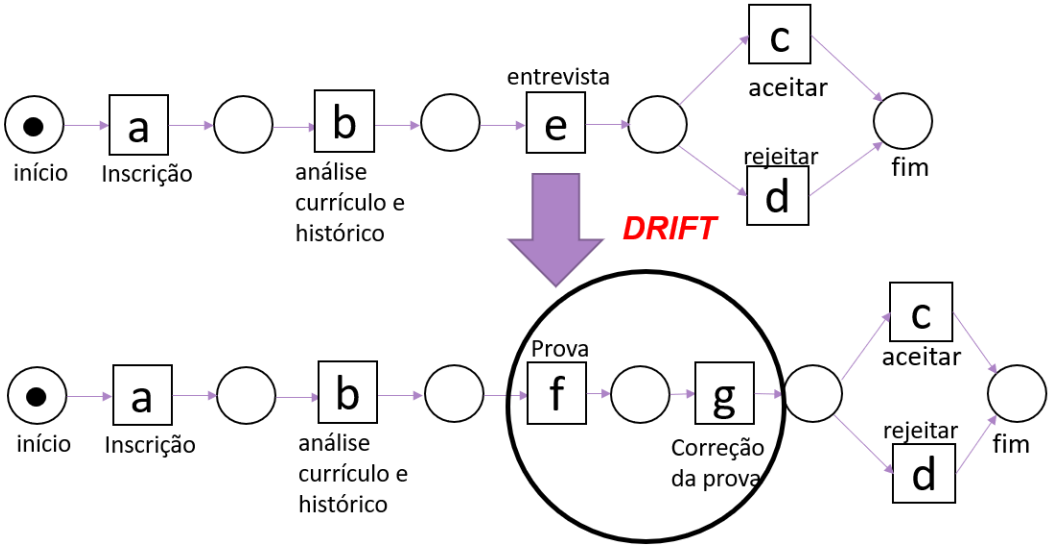
Fonte: Adaptado de Aalst (2016)

É fundamental ressaltar que as três perspectivas de mineração de processos estão intrinsecamente ligadas aos logs de eventos obtidos dos sistemas de informação. Os logs de eventos desempenham um papel crucial como entrada de dados para os algoritmos de mineração de processos, viabilizando a análise detalhada dos processos empresariais. Cada entrada em um log de eventos representa uma instância de processo, conhecida como caso, e contém informações essenciais como identificador do caso, atividade registrada e *timestamp*.

No entanto, é necessário salientar que a maioria dos algoritmos utilizados na descoberta de modelos de processos, conhecidos como mineradores, parte da premissa de que o log de eventos contém apenas uma versão do processo, desconsiderando possíveis alterações ao longo do tempo. Contudo, na prática, não pode-se presumir que um processo de negócio permaneça estático, pois ele está constantemente sujeito a influências como otimizações, variações na demanda, sazonalidades, eventos naturais adversos, crises e mudanças regulatórias (ZHENG; WEN; WANG, 2017). Em outras palavras, os processos de negócio estão em contínua evolução, seja de forma planejada ou imprevista.

Essas mudanças foram objeto de estudo por Bose et al. (2014), que investigaram uma ampla gama de processos em mais de 100 organizações, utilizando a ferramenta ProM,[1]. Seus resultados destacaram a distância entre a suposição de que os processos são estáticos e a realidade que ele estão passando por constantes mudanças.

Figura 2: Exemplo de *drift* no processo de negócio.



Fonte: O autor (2024)

Na literatura de mineração de processos, é reconhecida a presença de *concept drifts* ou *process drifts* em processos de negócios, denotando que tais processos podem sofrer alterações ao longo do tempo. Um exemplo ilustrativo é apresentado na Figura 2, representando um *drift* na estrutura de um processo seletivo fictício de pós-graduação de uma universidade. Nesse caso, a mudança na estrutura aconteceu devido a um aumento significativo no número de candidatos em comparação com o ano anterior, quando eram realizadas entrevistas de 30 minutos por candidato, optou-se por substituir essa etapa por uma prova objetiva presencial de uma hora para todos os candidatos.

No estudo conduzido por Zheng, Wen e Wang (2017), o *process drift* refere-se a modificação do processo durante sua execução. De acordo com Maaradji et al. (2015), o *process drift* é uma variação estatística nas propriedades da variável em análise. É fundamental reconhecer a complexidade dos processos de negócios, os quais envolvem estruturas de decisão, paralelismo e *loops*, tornando sua análise desafiadora.

[1] https://promtools.org/

Neste contexto, as abordagens atuais não se limitam apenas à detecção de *concept drift* segundo Zheng, Wen e Wang (2017), mas também se concentram em objetivos adicionais, como a detecção do ponto de mudança, a localização da mudança e a descoberta do processo de mudança (AALST, 2016). Em resumo, o objetivo não é apenas identificar que uma mudança ocorreu, mas também determinar o momento preciso em que ocorreu e situá-la dentro do modelo do processo, com o intuito de compreender a dinâmica do *drift*.

1.1 Problema Atual

Atualmente, uma variedade de ferramentas para mineração de processos já abordam a análise de processos em mudança. Entre elas, destacam-se:

- Zheng, Wen e Wang (2017) apresenta o TPCDD (*Tsinghua Process Concept Drift Detection*);
- Bose et al. (2014) apresenta o ProM – *Concept Drift Plugin*;
- Ostovar et al. (2016) apresenta o Apromore – *ProDrift Plugin*;
- Yeshchenko et al. (2019a) apresenta o VDD *system* (*Visual Drift Detection Tool*)
- Sato, Barddal e Scalabrin (2021) apresenta o IPDD (*Interactive Process Drift Detection*);
- Tavares et al. (2019) apresenta o CDESF (*Concept Drift in Event Stream Framework*).

Contudo, surgiu-se a dificuldade de encontrar uma ferramenta que possibilite a criação de logs de eventos com *drift* sintéticos próximos a vida real ao tentar comparar as diversas ferramentas disponíveis, conforme detalhado mais à frente na Tabela 1. Diante disso, surgem questionamentos importantes: Como é possível avaliar os métodos de detecção de *drifts* e compará-los usando bases de dados com *drifts* sintéticas? Quais registros/logs de eventos são mais adequados para validar a acurácia dos métodos? Será que o número de *drifts* influencia diretamente na precisão das ferramentas? Poderia a presença de muitos *drifts* nos logs diminuir a precisão das ferramentas? E, por fim, existem ferramentas disponíveis capazes de gerar logs de eventos contendo *drifts* com características específicas?

Na literatura, são escassas as bases sintéticas que oferecem *drifts* publicamente disponíveis, com exemplos como Maaradji et al. (2015), Ceravolo et al. (2020), Ostovar, Leemans e Rosa (2020). Essa escassez representa um desafio significativo para a validar e comparar algoritmos. Além disso, os geradores atualmente disponíveis exigem a combinação

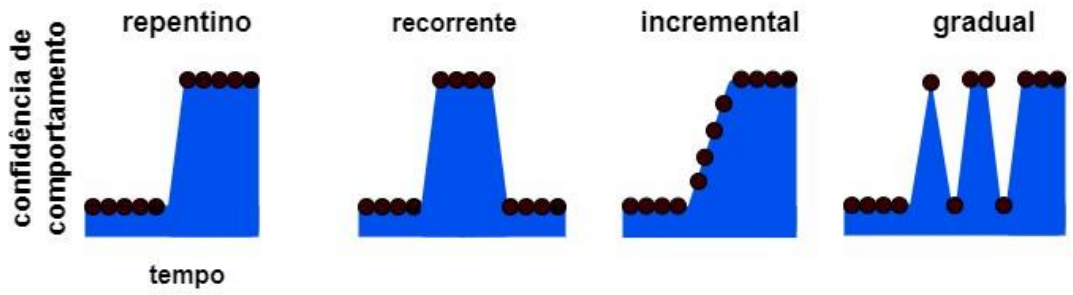
de várias técnicnas para produzir logs de eventos com *drifts* que incorporem os critérios essenciais necessários para a validação efetiva de ferramentas.

É essencial desenvolver um método que possibilite a incorporação de logs de eventos com diferentes tipos de *drifts* em um único conjunto de dados, a fim de validar adequadamente as ferramentas de detecção de *drift*. Segundo a classificação proposta por Bose et al. (2011), os *drifts* podem ser categorizados em quatro tipos distintos: abrupto (*sudden*), recorrente (*recurring*), gradual (*gradual*) e incremental (*incremental*). Portanto, a criação de conjuntos de dados em que apenas contemplem a previsão de *drifts* abruptos, como observado em estudos como Maaradji et al. (2015) e Ostovar, Leemans e Rosa (2020), pode não proporcionar uma representação fiel do comportamento de um processo real.

O VDD (*Visual Drift Detector*) *system*, Yeshchenko et al. (2020a), aborda os diversos tipos de *drift*, propondo uma abordagem em que o log de eventos é analisado com base em restrições, permitindo a detecção dos diferentes tipos de *drift*. Nele, testes estatísticos são realizados para a detecção de *drifts*, além da análise que é feita visualmente por meio de *drifts charts*, nos quais os *drifts* são classificados de acordo com os padrões exibidos na Figura 3.

Em Yeshchenko et al. (2020a) são gerados gráficos de autocorrelação para criar agrupamentos que permitem a detecção de logs de eventos com comportamento que representam *drifts* recorrentes. Essa mesma abordagem foi estendida por Yeshchenko et al. (2019b), Yeshchenko et al. (2020b), entretanto, a validação realizada não combina diferentes tamanhos de logs de eventos, diferentes intervalos entre os *drifts*, bem como combinações de tipos de *drift* no mesmo log de eventos .

Figura 3. Classificação visual dos drifts por meio de *drift charts.*



Fonte: Adaptado de Yeshchenko et al. (2020a)

Conforme delineado por Grimm, Kraus e Aa (2022), a avaliação e comparação eficazes de diferentes abordagens requerem logs de eventos que incorporem mudanças nos processo

(*process drift*), cujos detalhes previamente definidos e disponíveis como um padrão de referência (“*gold standard*”). Contudo, é incomum encontrar essa condição em logs de eventos do mundo real, acessíveis publicamente, nos quais apenas um número limitado de *drifts* é conhecido (GRIMM; KRAUS; AA, 2022). Os autores também observam que, no contexto de logs de eventos sintéticos, a falta de coleções de referência é evidente, e as ferramentas existentes para geração desses logs oferecem apenas suporte básico para incorporar *process drifts*, sem fornecer informações detalhadas necessárias para atender ao padrão de referência "*gold standard*".

1.2 Gerador de Logs e Protocolo de Experimentos

Diante deste panorama, a pesquisa visa desenvolver um método computacional para gerar conjuntos de dados compostos por logs de eventos sujeitos a *drifts*. Esses conjuntos de dados têm como propósito principal possibilitar uma ampla validação dos métodos atuais de análise de processos com *drifts*. A identificação da lacuna existente na área, especificamente a falta de um método computacional que simplifique a criação de bases sintéticas com *drifts*, é evidente. Portanto, o método que busca desenvolver a capacidade de gerar logs de eventos que se assemelham aos produzidos pelos sistemas de informação do mundo real.

Juntamente com o desenvolvimento do método para gerar logs de eventos sintéticos, é essencial a definição de um protocolo de teste fundamentado em análise estatística para avaliar a acurácia das ferramentas para detecção de *drifts*). Este protocolo deve empregar uma métrica de avaliação claramente definida, juntamente com parâmetros detalhados. O protocolo será aplicado a conjuntos de dados sintéticos, incluindo aqueles encontrados na literatura e aqueles gerados pelo método **PDG**. Essa iniciativa deve permitir verificar se há diferenças entre os métodos de geração de dados sintéticos.

Adicionalmente, em Sato et al. (2022) propõem a adoção do *F-score* para avaliar os algoritmos de detecção de *drifts* quando aplicados a logs sintéticos.

1.3 Objetivos geral e específicos

Desenvolver um método computacional para gerar conjuntos de sintéticas que contenham *process drifts* no formato de logs de eventos e sujeitos a diferentes critérios. E para a consecução deste objetivo geral foram definidos os seguintes objetivos específicos, a saber:

- **OE1**: Gerar conjuntos de logs de eventos contendo combinações de diferentes critérios para validar técnicas de detecção de *drifts*.
- **OE2**: Comparar a acurácia de diferentes métodos de detecção de *drifts* utilizando as bases de dados existentes na literatura e a gerada pelo método proposto.

Dados os subobjetivos em questão, o próximo passo do projeto foi a definição do método de pesquisa para a consecução deles.

1.4 Abordagem DSRM

O projeto de pesquisa adota a metodologia DSRM (***D**esign **S**cience **R**esearch **M**ethodology*), a qual segue uma sequência nominal de seis atividades distintas:

1. Identificação do problema e sua motivação.
2. Definição do objetivo para encontrar a solução do problema em questão.
3. *Design* e desenvolvimento da solução proposta.
4. Demonstração da solução.
5. Avaliação da solução desenvolvida.
6. Comunicação dos resultados obtidos.

A Tabela 1 apresenta as atividades/etapas e correlaciona com os capítulos do documento.

Tabela 1: Abordagem DSRM.

Atividade	Seção	Descrição
1	1.1 Problema Atual	Identifica a dificuldade para avaliar de forma comparativa ferramentas e gerar logs de eventos sintéticos com critérios adequados para compará-las
2	1.3 Objetivo e 2 Revisão Sistemática	Identifica os objetivos que foram delimitados a partir da revisão sistemática.
3	3. Método	Propõe o desenvolvimento de um gerador de log de eventos com *drift* e um protocolo de experimentos para comparar as ferramentas de detecção de *drift*.
4	4.1.1 Geração da Base Gerada 1, 4.2.1 Geração da Base Gerada 2, 5.1 Seleção e parametrização dos detectores de *drifts*	Reproduz logs de eventos existentes com a ferramenta. Cria conjuntos de dados sintéticos com novos parâmetros. Aplica o protocolo de análise estatística.
5	4.1.2Validação da Base Gerada 1, 4.2.2Validação da Base Gerada 2, 5.2 Análise dos resultados – comparação	Valida os logs de eventos replicados. Valida os logs de eventos criados. Analisa os resultados da comparação entre os detectores de *drift*.
6		Escrita da dissertação. Publicação de artigo, conjuntos de dados e código fonte para a comunidade científica.

Fonte: O autor (2024)

Nas próximas duas seções são apresentas a estrutura do documento e as contribuições esperadas com a realização desta dissertação.

1.5 Estrutura do Documento

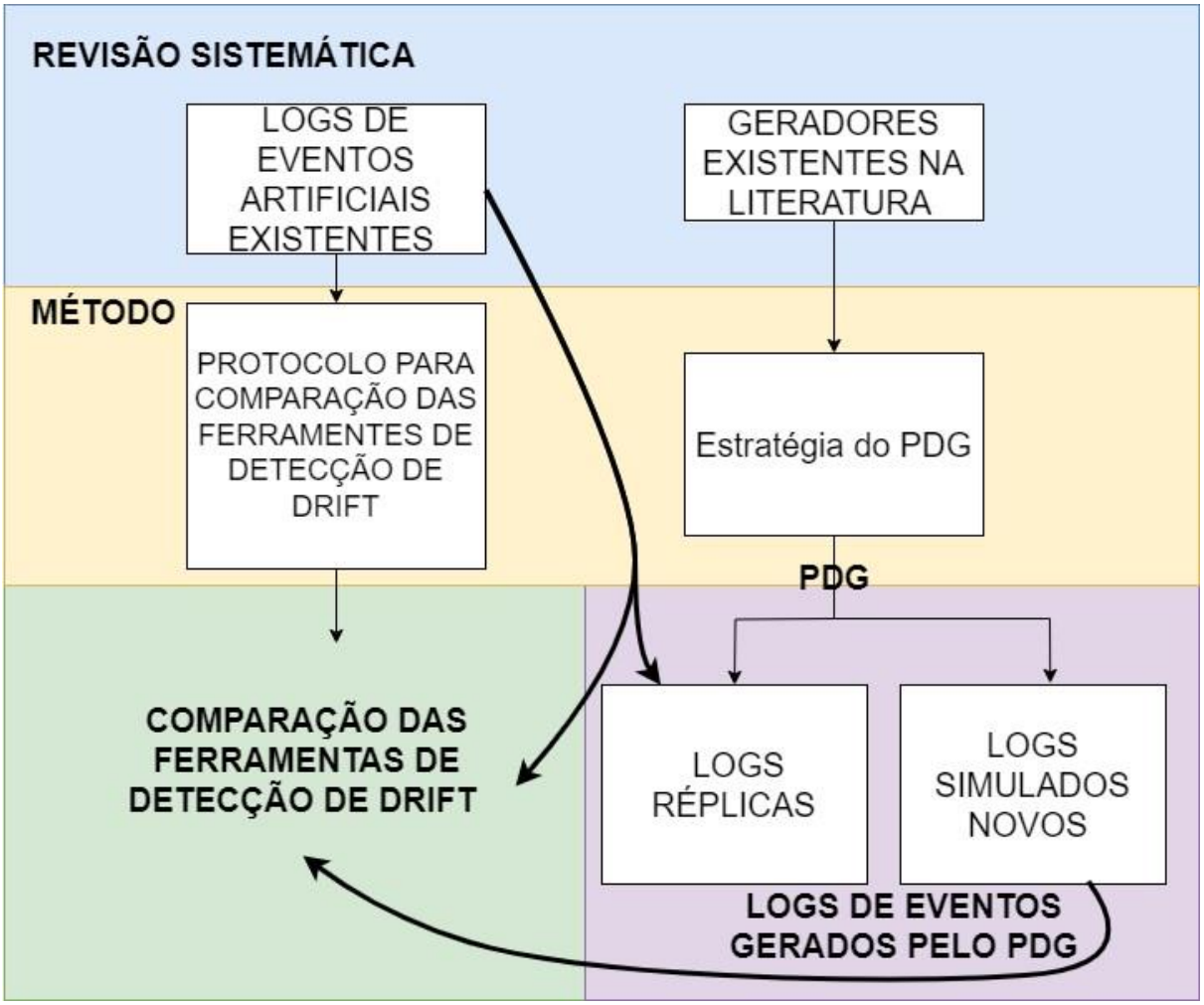
A estrutura do documento foi delineada de forma a iniciar com a apresentação da Revisão Sistemática (Capítulo 2), que abrange a identificação dos principais logs de eventos com drifts existentes, comparações entre ferramentas de detecção de *drift* e os geradores de logs de eventos. No Método (Capítulo 3), são detalhadas a estratégia de geração e validação dos logs de eventos adotada no **PDG**, bem como um protocolo estatístico para a comparação das ferramentas de detecção de *drift*. Seguindo adiante, no Capítulo 4 são delineadas duas bases de dados derivadas do **PDG**: uma consiste em logs replicados de uma base existente na literatura, enquanto a outra engloba logs de eventos com parâmetros novos. A validação dessas bases segue os métodos delineados no Capítulo 3, com destaque para a comparação entre a base replicada e a base de logs de eventos existente na literatura. No Capítulo 5, as ferramentas são submetidas à avaliação utilizando duas bases de logs de eventos existentes na literatura, bem como a base de logs de eventos gerada pelo **PDG**.

Essa estruturação é visualmente representada na Figura 4, proporcionando uma compreensão clara do fluxo de correlação entre os diferentes componentes do estudo.

O segundo capítulo deste trabalho propõe uma revisão sistemática com o objetivo de identificar, *por um lado*, a presença de geradores de logs de eventos artificiais com *drift* de processo e, *por outro lado*, a existência de conjuntos de dados artificiais e reais contendo logs de eventos também sujeitos a *drift* de processo. Simultaneamente, foi observada a falta de um protocolo de teste robusto para comparar métodos de detecção de *drifts* de forma clara e objetiva.

O terceiro capítulo sistematiza a geração de logs de eventos artificiais com *process drift* e define um protocolo de teste para a comparação de métodos de detecção de *drifts*. Tal protocolo é suficientemente geral, na medida em que ele pode ser aplicado para comparar o desempenho de detectores de *process drift*, tomando como entrada outras bases de dados, diferentes daquelas geradas pelo **PDG**.

Figura 4: Fluxo da estrutura do documento.



Fonte: O autor (2024)

O quarto capítulo apresenta a geração de uma base de dados e avaliação da corretude do método proposto **PDG**, denominada **base gerada 1**, a qual é uma réplica de uma base de dados da literatura; e finalmente, a geração de uma base de dados usando o *Process Drift Generator* (**PDG)**, denominada **base gerada 2**;

O quinto capítulo apresenta a comparação das ferramentas utilizando duas **bases de logs de eventos artificiais existentes** e a **base de logs de eventos gerada pelo PDG**. O sexto capítulo 6 apresenta as conclusões e os trabalhos futuros. De forma resumida, para que o leitor tenha um plano de leitura, o documento está estruturado da seguinte forma:

- Capítulo 2 (**Revisão sistemática da literatura**): buscas nas bibliotecas digitais, conceitos fundamentais, logs de eventos reais, bases de logs de eventos artificias existentes e geradores de logs de eventos existentes;

- Capítulo 3 (**Método**): apresentação da estratégia de geração e validação de logs de eventos pelo *Process Drift Gerenator*; apresentação do protocolo para comparação das ferramentas de detecção de *drift*;
- Capítulo 4 (**Logs de eventos gerados pelo PDG**): apresentação e validação de duas bases geradas sendo uma réplica de logs existentes e a outra logs com novos parâmetros;
- Capítulo 5 (**Comparação das ferramentas de detecção de *drift***): comparação das ferramentas de detecção de *drift* utilizando duas **bases de logs de eventos artificiais existentes** e uma **base gerada**;
- Capítulo 6 (**Considerações finais**): apresentação das conclusões.

1.6 Contribuições

As principais contribuições são: (a) uma revisão sistemática da literatura, identificando geradores de logs de eventos sintéticos e bases de dados com diferentes conjuntos de logs de eventos com *drifts* (artificias e reais); (b) a criação de um gerador de logs de eventos, permitindo a inserção de diferentes tipos de *drifts* e ruídos; e (c) sistematização de um protocolo de teste, permitindo a comparação das ferramentas de detecção de *drifts;* (d) uma base gerada pelo *Process Drift Generator* com diferentes parâmetros de entrada.

2 REVISÃO DA LITERATURA

A revisão sistemática da literatura foi abordada nesse capítulo da seguinte forma: na introdução apresenta-se as perguntas de pesquisa, a *string* de busca, as bibliotecas digitais utilizadas e a estatística dos artigos encontrados (Seção 2.1). Em seguida apresenta-se a taxonomia construída a partir dos artigos lidos (Seção 2.2) e na Seção 2.3 apresentam-se os resultados da revisão.

2.1 Introdução

O ponto de partida para iniciar a pesquisa foi a realização da revisão de literatura. Essa revisão teve como objetivo identificar as principais bases de dados com *process drifts* e quais as ferramentas utilizadas para a geração dessas bases caso elas sejam sintéticas. Dessa forma, foram postas três perguntas de pesquisa **(PPs**):

- **PP01**: Quais bases de dados de logs de eventos com *process drifts* inseridos sinteticamente e logs de eventos reais com *process drift* estão disponíveis para *download* na literatura?
- **PP02**: Quais abordagens providenciam uma ferramenta para a criação de base de dados de log de eventos com *drifts*?
- **PP03**: Quais foram os critérios de inserção dos *process drifts* nas bases de dados de log de eventos sintéticos encontrados a partir da PP01?

Dessa forma, a *string* de busca foi definida e a pesquisa foi feita em quatro bibliotecas digitais relevantes na área de pesquisa. A primeira busca foi feita em outubro de 2021 no início do projeto e para atualizá-la a mesma *string* foi executada em janeiro de 2024, para tal foi criado um programa para comparar os resultados das bases e reportar os novos artigos. O programa está disponível no github[2] em conjunto com os resultados das bibliotecas digitais (2021 e 2024) e pode ser aproveitado em outras revisões que precisem comparar saídas de dados das bibliotecas digitais. A Tabela 2 apresenta o número de resultados, a data de extração e o formato de texto da *string* por biblioteca.

[2] https://github.com/caioraduy/SystematicReview

Tabela 2: Buscas feitas em bibliotecas digitais.

Biblioteca	String de busca	Quantidade	Data de extração
Springer Link	("concept drift" | "process drift") & "process mining" & ("event log"| "event stream"| synthetic)	402	30/01/2024
ACM Digital Library	[[Full Text: "concept drift"] OR [Full Text: "process drift"]] AND [Full Text: "process mining"] AND [[Full Text: "event log"] OR [Full Text: "event stream"] OR [Full Text: synthetic]]	42	19/01/2024
IEEE Xplore	("Full Text & Metadata":"process drift" OR "Full Text & Metadata":"concept drift") AND ("Full Text & Metadata":"process mining") AND ("Full Text & Metadata":"event log" OR "Full Text & Metadata":"event stream" OR "Full Text & Metadata":synthetic)	79	30/01/2024
Scopus	(ALL ("process mining") AND ALL ("concept drift" OR "process drift") AND ALL ("event log" OR "event stream" OR synthetic))	397	30/01/2024

Fonte: O autor (2024)

Definiram-se os critérios de inclusão (**CIs**) utilizados para selecionar os artigos para a leitura *full-text*. Os **CIs** foram formulados para garantir as inclusões relevantes e os critérios de exclusão (**CEs**) para garantir a qualidade da seleção dos artigos de acordo com as **PPs**. Os artigos foram incluídos pela leitura do resumo.

- **CI01**: Artigos revisados por pares publicados em conferências ou periódicos em inglês que apresentem *event logs* ou *event streams* com *drift* para mineração de processos – 131 artigos
- **CI02**: Artigos revisados por pares publicados em conferências ou periódicos em inglês que abordam ou viabilizem a geração de *event logs* ou *streams* sintéticos com *process drift* - 8 artigos

Dessa forma, esses artigos foram lidos de maneira integral e filtrados perante os **CEs**:

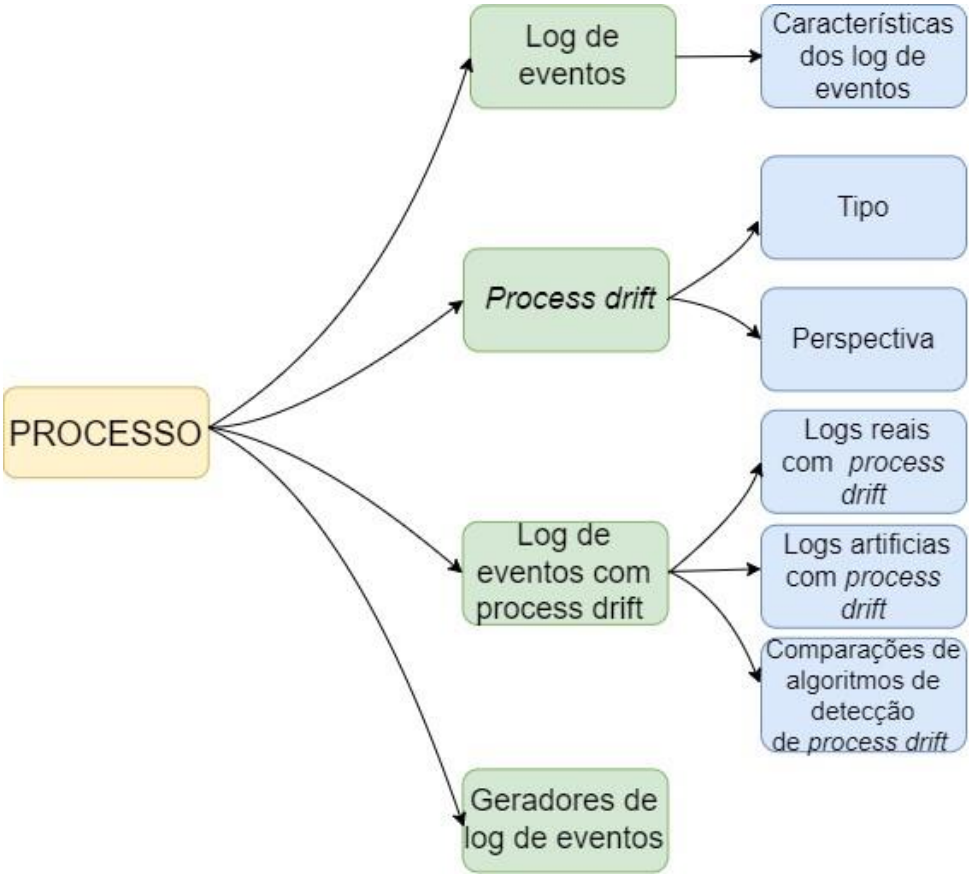
- **CE01**: Artigos duplicados – 42 artigos
- **CE02**: Artigos que apresentam uma ferramenta de detecção de *concept drift* ou predição considerando *drifts*, mas não introduzem nenhuma base de dados para teste – 6 artigos
- **CE03**: Artigos que não deixam clara como as bases de dados foram geradas e nem quais tipos de *process drifts* foram inseridos ou introduzem bases de dados apenas com inserção de anomalias ou sem detalhes sobre os *drifts* – 16 artigos.
- **CE04**: Artigo que apresentam *stream* de dados ou séries temporais como base de dados – 5 artigo.

Além disso, foram incorporados cinco artigos a partir da leitura da leitura *full-text*, conforme a técnica de *snowballing* apresentada em Wohlin (2014), que atendem ao **CI01**: Stocker e Accorsi (2013), Ratzer et al. (2003), Pourmasoumi et al. (2015), Hmami, Sbai e Fredj (2021) e Burattin e Sperduti (2011); totalizando 75 artigos.

2.2 Taxonomia da revisão

Os resultados foram divididos com base no esquema da Figura 5. Essa figura delineia uma taxonomia da revisão da literatura.

Figura 5: Taxonomia do conhecimento obtido na revisão sistemática da literatura.



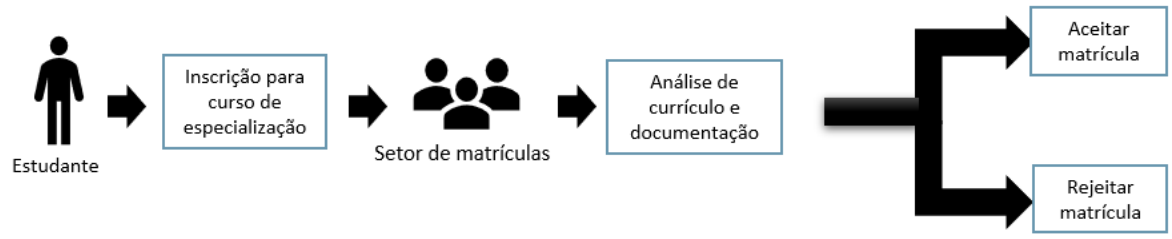
Fonte: O autor (2024)

2.3 Processo

Um processo consiste em uma sequência de atividades com um objeto final. Na Figura 6, pode-se observar uma simplificação de um processo seletivo em um curso de pós-graduação. O estudante faz a sua inscrição e envia a documentação – CPF, histórico escolar e currículo –

(atividade **a**), em seguida, o setor de matrículas faz a análise da documentação (atividade **b**) e, por último, o setor de matrículas aceita (atividade **c**) ou rejeita (atividade **d**) a matrícula do estudante.

Figura 6: Processo simplificado de seleção de um aluno em um curso de pós-graduação.

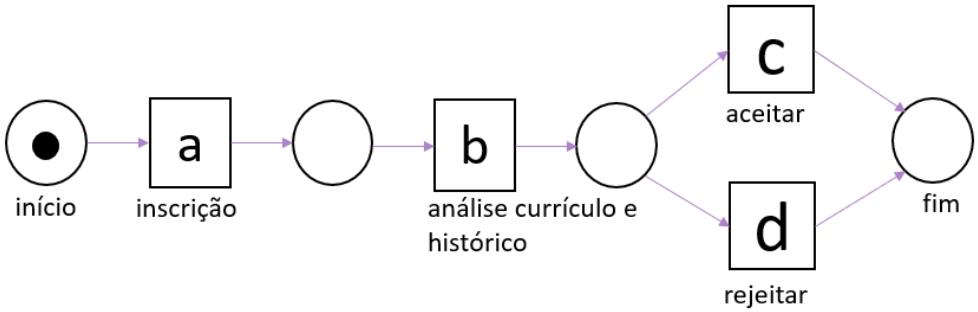


Fonte: O autor (2024)

Esses processos de negócios podem ser representados por meio de modelos de processo, que consistem na representação do processo que aconteceu na prática (em inglês, "*AS-IS"*).

A Figura 7 representa o modelo de processo de seleção de curso de pós-graduação em que existem dois caminhos, ilustrados aqui por dois traços: σ^1 =(a, b, c) e σ^2=(a, b, d). Uma instância do processo, ou caso, registra um traço no log de eventos. Esse processo pode ser ilustrado por meio de uma estrutura em rede de Petri (Figura 7), que é uma forma canônica de representação de um processo—na área de mineração de processos—, em que os quadrados (em inglês, "*transitions"*) indicam as atividades, conectadas pelos lugares (inglês, "*places"*). Para que uma atividade possa ser executada, todos os seus *places* de entrada devem possuir ao menos um *token*. Ou seja, se houver um *token* no primeiro *place* da rede de Petri, a atividade ***a*** está habilitada para a execução. Ao executar uma atividade, um *token* é consumido de cada *place* de entrada e um *token* é produzido em cada *place* de saída. Ao executar a atividade ***a***, por exemplo, um *token* é produzido no segundo *place*, habilitando a execução da atividade **b** (AALST, 2016).

Figura 7: Modelo de processo do processo seletivo de curso de pós-graduação.



Fonte: O autor (2024)

2.3.1 Logs de eventos

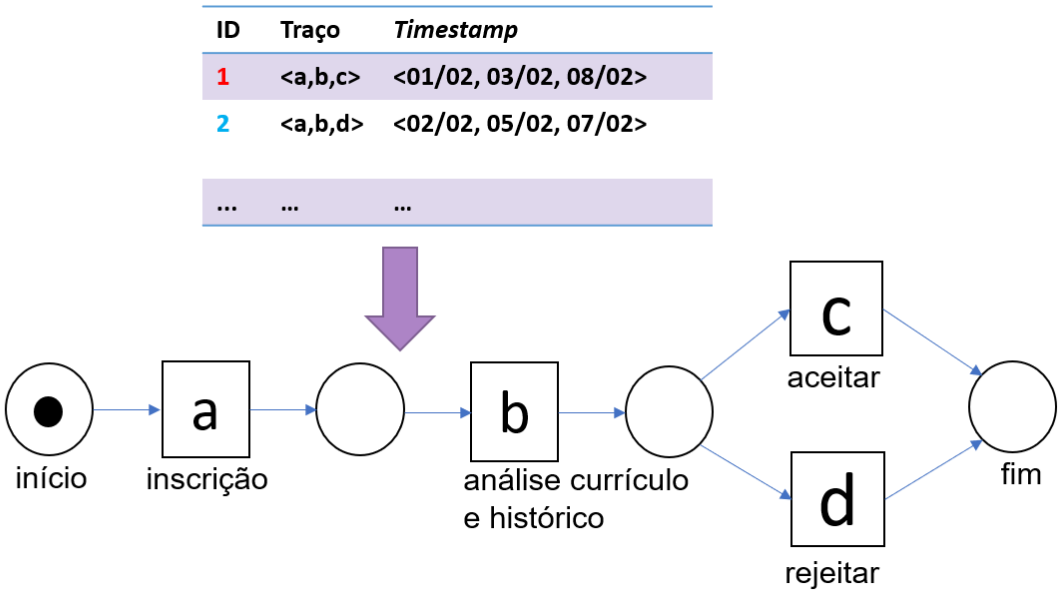
Um log de eventos contém informações sobre todos os eventos registrados para um processo de negócio. A Tabela 3 representa o log de eventos (*L*) do processo seletivo do curso de pós-graduação. A primeira coluna representa o identificador do caso, dessa forma, o *Caso 1* representa a sequência de atividades σ^1=(a, b, c) e o *Caso 2* representa a sequência σ^2=(a, b, d). No exemplo, o identificador representa um estudante que foi submetido ao processo. Os algoritmos de descoberta, cf. ilustrado na Figura 8, extraem o modelo de processo a partir dos traços (σ). Nela, observa-se acima os possíveis traços e abaixo a representação de modelo de processo usando a notação de Rede de Petri. Um log de eventos com mais atributos permite analisar os processos em outras perspectivas, por exemplo, tempo e recursos (AALST, 2016).

Tabela 3: Exemplo hipotético de um log de eventos do processo de seleção de estudantes em um curso de pós-graduação.

ID	Atividade	Abreviatura	*Timestamp*	Responsável
1	inscrição	a	21/10/2020 10:50	Pedro
1	análise currículo e histórico	b	23/10/2020 09:00	Maria
1	aceitar	c	22/10/2020 09:15	Carla
2	inscrição	a	22/10/2020 14:20	Pedro
2	análise currículo e histórico	b	22/10/2020 17:10	Maria
2	rejeitar	d	23/10/2020 14:00	João
…	...	...	...	…

Fonte: O autor (2024)

Figura 8: Extração do modelo de processo a partir dos traços do log de eventos.



Fonte: O autor (2024)

Em resumo, um log de eventos deve conter no mínimo três atributos: identificador do caso, atividade executada e o *timestamp* (marca temporal). As sequências de atividades, dão origem aos traços. Portanto, a leitura de um log de eventos pode ser feita pela ordenação dos traços ou dos eventos pela marca temporal. Deve-se enfatizar que os logs de eventos são usados para testar algoritmos que trabalham de maneira *offline* (AALST, 2016).

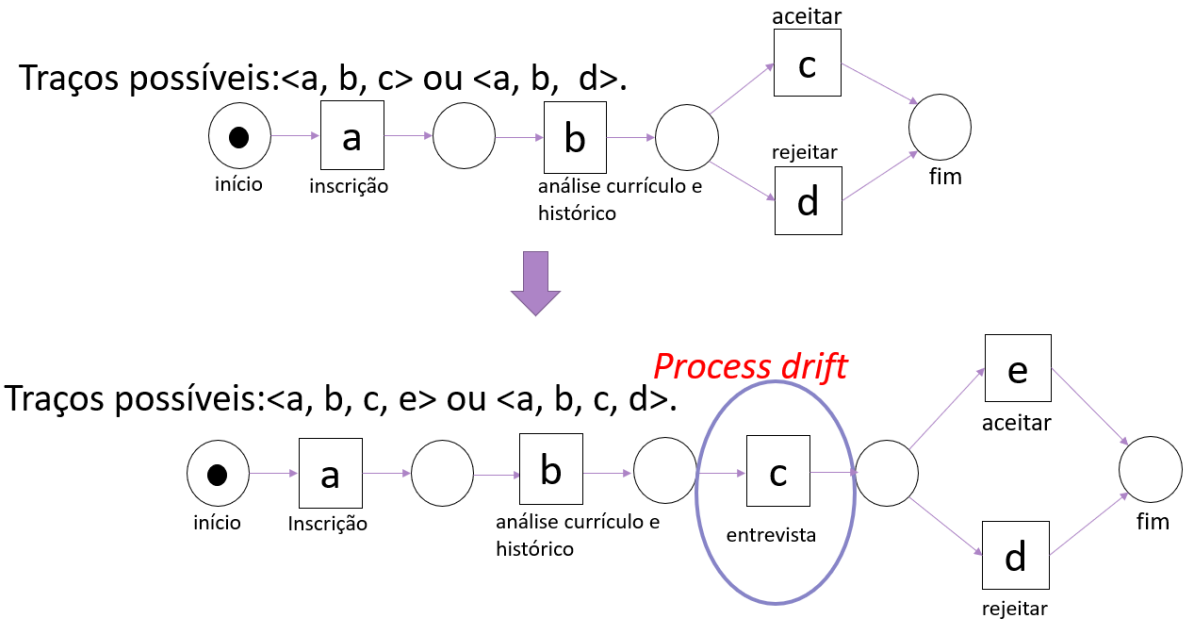
Características de log de eventos

Um log de eventos tem características específicas tais como os atributos mínimos (identificador do caso, atividade executada e o *timestamp*) que o diferenciam de um conjunto de dados clássico e de uma série temporal. Cada log de eventos pode ser armazenado em CSV, XES ou MXLM, ou outros formatos. Além disso, todo log de eventos pode ter atributos adicionais que possibilitam uma análise com mais detalhes sobre o processo, como por exemplo, diferentes categorias de recursos. Já o tamanho do log depende do número de traços e eventos, quanto maior o número desses, maior será custo computacional face à execução de cada tarefa de mineração de processos. Além disso, os logs podem conter ruídos e/ou mudanças no processo em análise, denominado *process drift* ou *concept drift*.

2.3.2 Process Drift

A definição de *concept drift* é utilizada na mineração de dados, sendo primeiramente mencionada em Schlimmer e Granger Jr (1986). *Concept drift* refere-se as mudanças na saída resultante do algoritmo causada por mudanças nos dados de entrada (RICHTER; SEILD, 2017). Os autores em Bose et al. (2011, 2014) definiram as mudanças nos processos de negócios que ficaram conhecidas como *process drift*. O *process drift* também tem uma relação com a entrada de dados – o log de eventos – e a saída – o modelo do processo. A mineração de processos usa estruturas complexas (concorrência, escolha e repetições) no modelo de processo que impossibilitam que os algoritmos de mineração de dados sejam utilizados diretamente. As técnicas de mineração de dados usam vetores com dados categóricos ou contínuos como entrada e tem como saída relações entre esses dados (SATO et al., 2022). A Figura 9 ilustra uma situação em que um processo de seleção de estudantes sofre uma mudança. A mudança ocorre quando é inserida a atividade "entrevista".

Figura 9: Modelo de processo seletivo de curso de pós-graduação com *process drift*.

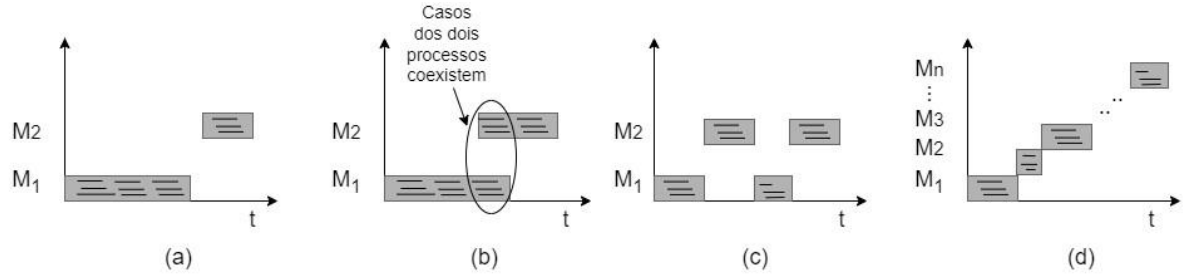


Fonte: O autor (2024)

Tipo

Quatro tipos de *drifts* foram identificados na literatura por Bose et al. (2014): abrupto, gradual, recorrente e incremental. A Figura 10 ilustra graficamente esses quatro tipos de *drifts*.

Figura 10: Tipos de *drifts*.



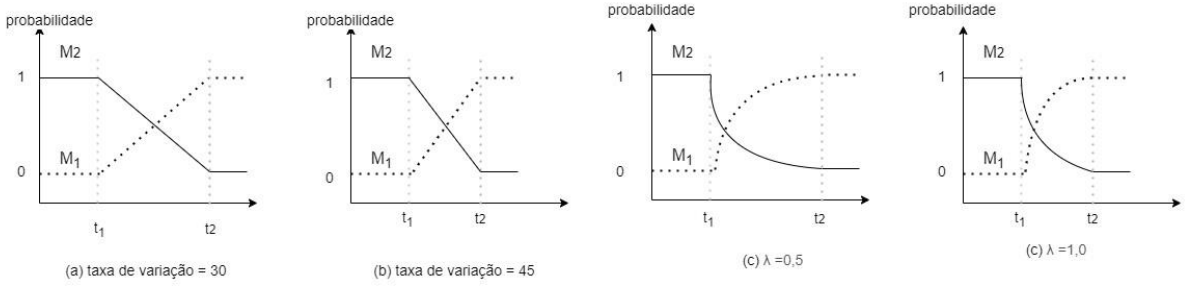
Fonte: Adaptado de Bose et al. (2014)

Em Bose et al. (2014), define-se o *drift* abrupto (a) como a substituição abrupta/repentina de um processo existente m_1 por um processo m_2: no momento em que m_2 começa a existir, m_1 deixa de existir. Ou seja, a partir desse momento todas as instâncias do processo m_1 deixam de existir. Isso ocorre normalmente em situação emergencial, por exemplo, quando um novo decreto que interfere no funcionamento de um aeroporto (BOSE et al., 2011).

Ainda em Bose et al. (2014), um *drift* gradual (b) consiste em um cenário no qual um processo atual m_1 é substituído por m_2, mas ambos coexistem por algum tempo até m_2 substituir o modelo de processo m_1. Isso ocorre, por exemplo, em uma organização de cadeia de suprimentos, em que a introdução de um novo processo no *delivery*, faz com que as ordens sigam o novo processo apenas após essa nova determinação (BOSE et al., 2014). Assim, as ordens emitidas anteriormente continuam sendo executadas conforme o modelo anterior.

A Figura 11 representa diferentes *drifts* graduais mostrando como as probabilidades dos modelos (eixo y) diminuem/aumentam ao longo do tempo (eixo x). Em (a) e (b) são apresentados *drifts* graduais com decaimento linear, onde as probabilidades aumentam/decaem de maneira continua. A taxa de variação define a intensidade em que o *drift* será substituído e de forma indireta o número de traços que durará o *drift*, isso porque após t_2 todos os traços passam a ser gerados a partir de M_2. Em (c) e (d) são exemplificados *drifts* graduais com decaimento exponencial em que M_1 segue a função $P[M_1] = e^{-\lambda t}$ e M_2 segue $P[M_2] = (1 - P[M_1])$ (Martjushev, Bose e Aalst, 2015 apud Sato et al., 2022).

Figura 11: Funções de decaimento em *drifts* graduais.



Fonte: Martjushev, Bose e Aalst (2015 apud Sato et al., 2022)

O *drift* recorrente (c) consiste em um modelo de processo que reaparece após um intervalo de tempo. Isso ocorre em um processo que é influenciado pela sazonalidade. Por exemplo, uma agência de viagens tem um aumento de suas vendas em períodos de férias escolar e natal. Essa recorrência pode ser periódica ou não, por isso é de difícil análise (BOSE et al., 2014).

O último tipo de *drift* é o incremental (d), em que um processo m_1 é substituído por outros processos m_n com pequenas mudanças incrementais (BOSE et al., 2014). Isso pode ocorrer, por exemplo, em organizações que passam por uma sequência de pequenos aprimoramentos na qualidade do gerenciamento.

Além disso, em Sato et al. (2022) considera-se os *drifts* graduais e abruptos como padrões básicos de mudanças. E, os *drifts* incrementais e recorrentes podem combinar *drifts* graduais, abruptos ou ambos. Em Xu, Zhang e Duan (2023), os autores apresentam o conceito "*nested drift*" que refere-se a *drifts* repentinos que ocorrem dentro do período de transição de um *drift* gradual, mostrando que os *drifts* são aninhados e interdependentes.

Perspectiva

Os *process drifts* são classificados face as perspectivas que afetam o modelo de processo: fluxo (sequência de atividades), tempo, comportamento, dados e recurso. O *drift* mais comum encontrado na literatura é o na perspectiva de fluxo, em que existe uma alteração na estrutura do processo. A Figura 9 representa um *drift* na perspectiva de fluxo, onde os traços são alterados. Alguns autores se referem aos *drifts* na perspectiva de fluxo como *drifts* estruturais, como é o exemplo em Richter e Seidl (2017). Assim, ao longo do documento, *drifts* estruturais e *drifts* na perspectiva fluxo são tratados como sinônimos.

Um *drift* na perspectiva temporal considera a marca temporal de cada registro, podendo ser uma mudança no tempo de duração de uma atividade ou uma alteração no intervalo entre atividades.

Em Yang et al. (2022a), os autores definem "*behavioural drift*" como sendo mudanças em que a estrutura de um processo que permanece a mesma, mas os padrões ou trajetos dos clientes ao longo do processo são significativamente diferentes. Por exemplo, os clientes podem começar a trazer suas próprias sacolas em vez de comprar uma sacola plástica quando vão às compras ou, devido a um fator repentino, como a COVID-19, os clientes se tornam mais propensos a obter produtos frescos por meio de entrega em vez de ir à loja (YANG ET AL., 2022a). Além disso, em Gallego, Vidal e Lama (2023) classifica como "*behavioural drift*" padrões de mudanças em que existe a variação da frequência de uma variante/caminho de um processo.

Deve-se salientar que há vários estudos alinhados aos detectores dos diferentes tipos/perspectivas de *drifts*: Richter e Seild (2017), Richter e Seidl (2019), Che, Machu e Zhou (2016), Brockhoff, Uysal e Aalst (2020). Em Stertz e Rinderle-Ma (2019) busca-se detectar *drift* nos dados/atributos do processo; aqui, os autores salientam a importância de identificar os *data drifts* em tempo real para conseguir manter o controle dos processos. Deve-se notar que isso não é trivial quando a análise é feita após a execução do processo. Dessa forma, defende-se que os *data drifts* devem ser detectados em tempo real, ou seja, *online*. Por exemplo, em um processo de logística em que os atributos são: marca temporal, nome do evento, recurso e velocidade média de entrega. Quando o atributo velocidade média tem uma diminuição significativa, tem-se um *data drift*. Isso pode ser causado por múltiplas razões como obras na estrada, entre outros motivos (STERTZ; RINDERLE-MA, 2019).

Deve-se notar também que os *drifts* também podem ocorrer na perspectiva de recurso, como por exemplo, uma mudança no setor responsável por uma atividade.

2.3.3 Logs de eventos com process *drift*

Deve-se notar que um dos objetivos de um algoritmo de detecção de *drifts* é identificar mudanças não esperadas. Portanto, para que esse seja testado faz-se necessário ter a garantia da ocorrência de um *drift*, o que dificulta a utilização de logs reais. Para validar um *drift* em um processo real é necessário que um especialista indique sua existência. Dessa forma, embora existam logs de eventos reais com *process drift*, a geração de logs sintéticos com *drifts*, inseridos artificialmente, é uma prática que permite uma avaliação mais objetiva dos

algoritmos/métodos dedicados a detecção de *drifts*. A avaliação dos algoritmos, por métricas como *recall*, *precision* e *F-score*, conforme Powers (2020), necessita que a localização de cada *drift* seja precisamente conhecida.

Logs de eventos reais com *process drift*

Foram encontrados na literatura logs de eventos reais com *process drift*. No entanto, nem todos os logs se encontram disponíveis para acesso. A Tabela 4 resume apenas os logs de eventos reais (abreviados como E na tabela) disponíveis para a comunidade científica indicando diferentes características. Nela, os *drifts* encontrados nos logs de eventos no artigos são reportados, no entanto, e assim não são "*gold standards*" que possam vir a ser utilizados na validação das ferramentas para a detecção de *drift.*

Tabela 4: Logs de eventos reais da literatura.

Log de eventos reais	Modelo de processo	Número de possíveis *drifts*	Perspectiva dos *drifts*	Número de eventos	Número de traços	Localização do *drift*	Tipo de *drift*	Utilizado em
E1[3]	Processo de aplicação de empréstimos (BPIC 2017)	2	Tempo e Recurso	1.202.267	4 047	Semana 28-> Perspectiva: Tempo Semana 21-> Perspectiva: Recurso	Abrupto	Adams et al. (2021), Spenrath e Hassani (2020), Adams et al. (2023a)
E2[4]	Faturamento de serviços médicos	2	Fluxo	451.359 eventos anonimizados	100.000	Case 71 e case 91	Abrupto	Barbon Junior et al. (2018), Yang et al. (2021), Yang et al. (2022a)
E3[5]**-5 logs**	Reembolso de viagens da universidade TU/e (BPIC 2020)							Lu, Chen e Poon (2021a), Sousa et al. (2024), Huete, Qahtan e Hassani (2023), Lin et al. (2023)
E3-1	Pedidos para pagamento	2	Fluxo	36.796	6.886	(2018-01-06 20:00:55) e (2017-12-22 02:56:06)	Abrupto	Lu, Chen e Poon (2021a)
E3-2	Declarações de viagens domésticas	2	Fluxo	56.437	10.500	(2018-01-06 19:42:04) e (2017-12-22 22:07:12)	Abrupto	Lu, Chen e Poon (2021a)
E3-3	Custos de viagens pré-pagas	2	Fluxo	18.246	2.099	(2018-01-07 02:22:19) e (2017-12-19 19:22:00)	Abrupto	Lu, Chen e Poon (2021a)
E3-4	Declarações de viagens internacionais	2	Fluxo	72.151	6.449	(2018-01-06 21:13:26) e (2017-12-22 03:11:38)	Abrupto	Lu, Chen e Poon (2021a)
E3-5	Autorizações de viagens	2	Fluxo	86.581	7.065	(2018-01-06 21:13:26) e (2017-12-22 03:11:38)	Abrupto	Lu, Chen e Poon (2021a)
E4[6]	Tratamento de pacientes em um hospital holandês (BPIC 2011)	2	Fluxo	150.291	981	(06/09/2007-evento 71.321), (29/11/2007-	Abrupto	Hompes et al. (2015), Ostovar et al. (2016), Hompes et al. (2017),

[3] https://www.win.tue.nl/bpi/doku.php?id=2017:challenge
[4] https://data.4tu.nl/repository/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741
[5] https://icpmconference.org/2020/bpi-challenge/
[6] http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

						evento 78.541), 12/07/2005, 21/08/2006		Lin et al. (2020), Burattin et al. (2015), Maggi et al. (2013), Yeshchenko et al. (2021)
E5[7]	Gerenciamento de *tickets* de uma central ajuda de uma empresa italiana	3	Fluxo	21 438	4 580	(2011/01/19), (2011/07/26), (2012/08/10)		Ostovar, Leemans e Rosa (2020), Lu, Chen e Poon (2021b)
E6[8]**-5 logs**	Aplicações de permissão de construção em 5 municípios holandeses (BPIC 2015)							Zellner et al. (2020), Hompes et al. (2015), Hompes et al. (2017), Hassani (2019)
E6-1	Município 1	1	Fluxo	52.217	1.170	31/08/2012 e 20/04/2013	Abrupto	Zellner et al. (2020), Hompes et al. (2015), Hompes et al. (2017), Hassani (2019), Hanga, Kovalchuk e Gaber (2022), Huete, Qahtan e Hassani (2023)
E6-2	Município 2	---	---	44.354	828	---	---	Hassani (2019)
E6-3	Município 3	---	---	59.681	1.349	---	---	Hassani (2019)
E6-4	Município 4	---	---	47.293	1.049	---	---	Hassani (2019)
E6-5	Município 5	---	---	59.083	1.156	---	---	Hassani (2019)
E7[9]	Processo de sistema de gestão de multas de trânsito	1	Temporal	561.470	231	Toda primavera	Recorrente	Richter e Seidl, (2019)
E8[10]	Processo de solicitação de empréstimo pessoal ou cheque especial dentro de uma organização financeira global (BPIC 2012)	____	___	262.200	13.084	___	___	Maggi et al. (2013)

[7] https://data.4tu.nl/articles/conjunto de logs de eventos/Conjunto de logs de eventos_belonging_to_the_help_desk_log_of_an_Italian_Company/12675977/1
[8] https://data.4tu.nl/collections/_/5065424/1
[9] https://data.4tu.nl/articles/conjunto de logs de eventos/Road_Traffic_Fine_Management_Process/12683249/1
[10] https://www.win.tue.nl/bpi/doku.php?id=2012:challenge&redirect=1id=2012/challenge

E9[11]	Incidentes e problemas de um sistema de gerenciamento chamado VINST (BPIC 2013)	___	___	74.544		____	___	Maggi et al. (2013)
E10[12]	Casos de sepse em um hospital	2	Fluxo	15.214	1.050	(05-25-2014), (09-24-2014)	Abrupto	Yeshchenko et al. (2020a), Yeshchenko et al. (2021), Guan et al.(2023)
E11[13]	Licença de construção em cinco municípios anônimos		Fluxo	8.577	1.434	Traços: 202, 894	Abrupto	Guan et al. (2023)

Fonte: O autor (2024)

[11] https://www.win.tue.nl/bpi/doku.php?id=2013:challenge
[12] https://data.4tu.nl/articles/conjunto de logs de eventos/Sepsis_Cases_-_Event_Log/12707639
[13] https://data.4tu.nl/articles/_/12709127/1

Log de eventos real 1 (E1): No estudo conduzido por Adams et al. (2021), apresenta-se e testa-se um *framework*, e relata-se *drifts* na perspectiva de *performance* (temporal — uma vez que uma mudança na *performance* considera o tempo de serviço) na semana 28. A média de duração da atividade "*W_Validate application*" tem uma diminuição significativa na semana em questão. Já na semana 21, observou-se um aumento na carga de trabalho dos recursos do processo (*drift* na perspectiva de recurso). No referido estudo, constatou-se que o aumento na carga de trabalho dos recursos levou a uma diminuição da duração da atividade "*W_Validate application*" (ADAMS et al., 2021).

No estudo conduzido por Adams et al. (2023a), que é uma extensão do estudo realizado em Adams et al. (2021), identificou-se novamente um aumento na carga de trabalho na semana 20, seguido por uma diminuição nos tempos de serviço na semana 25. Na análise da relação entre esses dois eventos, encontrou-se novamente uma conexão importante, já apontada em Adams et al. (2021): o crescimento na carga de trabalho influenciou diretamente a redução nos tempos de serviço. Já em Spenrath e Hassani (2020), não relata-se se há ou não *drifts* no log de eventos.

Log de eventos real 2 (E2): O log de eventos "*Hospital Billing*" foi obtido do subsistema financeiro do sistema ERP de um hospital, contendo eventos relacionados à cobrança de serviços médicos. Cada caso presente no log de eventos documenta as atividades realizadas em relação a cobrança de um pacote de serviços médicos. O log de eventos contém 100.000 traços coletados ao longo de três anos com os seguintes atributos: estado do processo, tipo do caso e diagnóstico. Deve-se notar que embora os casos tenho sido anonimizados, mantendo a ordem temporal dos eventos dentro de cada rastro, os *timestamps* são aleatórios. Os registros presentes no log de eventos são de 2012 até 2016 (MANNHARDT, 2017). Em Barbon junior et al. (2018), diferentes parâmetros do *framework* foram exercitados, o que permitiu encontrar *process drifts* no caso 71 e caso 291. Os casos deixaram de ser classificados como *outliers* e se tornaram padrões.

Em Yang et al. (2022a), ao comparar as probabilidades de transição, foram detectados seis *drifts* repentinos. Os *drifts points* são rotulados com as letras de A até F, localizados nos índices 4700, 5900, 8700, 16.300, 60.700 e 70.300. Além disso, foram encontrados adicionalmente 58 *drifts* em outros atributos que foram filtrados no processo de localização e racionalização dos *drifts*, por meio de uma análise mais aprofundada do log de eventos e dos *drifts* reportados.

Em Yang et al. (2021), fez-se uma análise na duração dos processos dos pacientes, desde a fase inicial ("*New*") até a conclusão ("*Billed*"). Essa análise foi realizada dividindo os casos em cinco subperíodos, com base no momento de início da aplicação (2013a, 2013b, 2014a, 2014b e 2015a), e aplicando um modelo de mistura de distribuições *gamma* para modelar a duração dos processos dentro de cada subperíodo. Para inspecionar diferenças entre períodos consecutivos, foram utilizados histogramas, distribuições de densidade e cumulativas, além de métricas de avaliação. Os *drifts* foram identificados entre os períodos de 2013a e 2013b, e entre 2014b e 2015a, marcados por alterações significativas nas distribuições das durações dos processos. Especificamente, as principais diferenças entre 2013a e 2013b foram observadas em durações específicas do processo, a saber: 73.5, 103, 133, 162.5 e 192 dias. A utilização de métricas estatísticas, incluindo o teste de *Kolmogorov-Smirnov*, confirmou, segundo os autores Yang et al. (2021), os *drifts* significativos nesses períodos.

Log de eventos real 3 (E3): Os logs do BPIC 2020 foram todos referentes ao processo de reembolso da *University of Technology* (TU/e) de 2017 a 2018. Cada log encerra a um tipo de pedido de reembolso e todos possuem um fluxo similar. Em Lu, Chen e Poon (2021a) foram encontrados *drifts* diferentes entre o modelo no final do ano de 2017 e no começo do ano de 2018. Em outras palavras, constatou-se a existência de um *drift*: onde, antes um documento submetido para o reembolso era classificado como documento pré-aprovado ou enviado para o supervisor responsável para aprovação, e, na sequência, o documento passou a ser direcionado para administração analisar e, se aprovado, enviado para o supervisor dar prosseguimento no processo de reembolso (LU; CHEN; POON, 2021a).

Em Sousa et al. (2024), ressalta-se que mesmo sem uma confirmação definitiva da presença de *drifts* nos logs de eventos analisados, identificou-se evidências desses *drifts* no processo de negócios entre o final de 2017 e o início de 2018. Acredita-se que a digitalização do processo, que passou a exigir o preenchimento digital de todos os casos, pode ter sido a causa, explicando assim o aumento no volume de casos nesse intervalo. Adicionalmente, destaca-se que a coleta de dados até o final de 2017 se limitava a apenas dois departamentos, enquanto a partir do início de 2018 expandiu-se para incluir todos os departamentos. Essa expansão nas fontes de dados pode ter contribuído para a detecção dos *drifts*, considerando que os diversos departamentos podem apresentar métodos de trabalho distintos. Em Huete, Qahtan e Hassani (2023), afirma-se que a expansão do processo para múltiplos departamentos é um

drift gradual, e, com base nessa premissa, comparam o detector proposto com existentes na literatura. Lin et al. (2023) também reporta o *drift* entre 2017 e 2018, mas não o caracteriza.

Log de eventos real 4 (E4): Em Ostovar et al. (2016), detectou-se *drifts* abruptos na perspectiva de fluxo nos eventos: 71 321 (06/09/2007) e 78 541 (29/11/2007), de um processo de um hospital acadêmico holandês. Já em Hompes et al. (2015), o mesmo processo foi comparado entre 2005 e 2006. E, em 2006 foram adicionadas mais partes diagnosticadas do corpo (doenças para partes do corpo que não eram registradas no processo) com os códigos M13, 822 e 106. Observou-se também que os tratamentos para doenças específicas mudaram durante o tempo, novos diagnósticos e doenças foram adicionadas Hompes et al. (2015). Em Hompes et al. (2017), constatou-se um potencial *drift* em Julho de 2006, quando houve uma mudança no comportamento dos *clusters*; os diagnósticos se tornaram mais específicos.

Em Lin et al. (2020) usou-se vários detectores de *process drifts* visando encontrar um *drift* em comum. Eles encontram um *drift* em torno do traço 200. Em Maggi et al. (2013) e Burattin et al. (2015) usaram-se o mesmo log apenas para validar o processamento do algoritmo proposto. E, em Yeshchenko et al. (2021) foram encontrados dois *drifts* abruptos na segunda metade do tempo de *span* do log (12/07/2005 e 21/08/2006) e *drifts* incrementais em dois agrupamentos gerados pela estratégia proposta (grupo 15 e grupo 18). Esses grupos foram gerados por meio da aplicação de algoritmos de agrupamento e, portanto, não podem ser usados como um marcador de localização no log de eventos.

Log de eventos real 5 (E5): Em Ostovar, Leemans e Rosa (2020) encontrou-se um *drift* de frequência no dia 31 de Agosto de 2012, o qual também foi encontrado em Lu, Chen, Poon (2021b). Deve-se notar que também se encontrou um *drift* no dia 20 de Abril de 2013.

Log de eventos real 6 (E6): Os logs do BPIC 2015 são referentes as permissões de construções de cinco municípios holandeses (M1, M2, M3, M4 e M5), cada log é referente a um munícipio. Em Hompes et al. (2015), relata-se dificuldades em interpretar a mudança nos *clusters* formados entre os anos de 2011 e 2012 no log E6-1. É importante citar que foram detectados *drifts* por meio da comparação de *clusters*. Em Hompes et al. (2017) apontou-se que um possível *drift* ocorreu no dia 13 de Janeiro de 2013 no mesmo log. E em Zellner et al. (2020) encontrou-se um *drift* do tipo incremental nos registros do Município 1, mas não ficou claro qual era a localização do *drift* e quais foram as alterações ocorridas. Já em Hassani (2019), Hanga, Kovalchuk e Gaber (2022) e Huete, Qahtan e Hassani (2023) juntaram-se os 5 logs para

gerar um log com modelos diferentes, ou seja, com 4 *drifts*. Logo, para cada transição de um log para outro, criou-se um *drift*.

Em Sousa et al. (2024), encontrou-se um *drift* no final de 2012 nos municípios M1, M3, M4 e M5, e um *drift* no primeiro semestre de 2013 no município M2. Esses *drifts* foram relacionados as ações de tomada de decisão – "*Phase Decision Ready*", "*Phase Draft*" "*Decision Ready*" e "*Phase Decision Sent*". No município M1, o *drift* foi detectado precocemente e a atividade "*Phase Case Handled*" também é apontada como uma atividade provavelmente envolvida no *drift* (SOUSA et al., 2024). Adicionalmente, em Sousa et al. (2024), afirmou-se que os experimentos relatados corroboram com a hipótese da existência de *process drift* no log de eventos, mas que tais resultados precisam ser validados por especialistas do domínio a fim de se obter respostas definitivas.

Log de eventos real 7 (E7): O log de eventos (E9) contém em torno de 150 mil casos. Esse log não tem *drifts* estruturais (RICHTER; SEIDL, 2019). Por outro lado, estão presentes *drifts* recorrentes na perspectiva temporal: o tempo entre as atividades "*Insert Fine Notification*" e "*Add Penalty*" reduz durante toda primavera e aumenta na metade do ano. Isso foi causada pelo horário de verão que alterou as marcas temporais dos registros, gerando *drifts* artificiais. A padronização do atributo tempo, nesse caso, evitaria que ocorresse um *drift* temporal no log (RICHTER; SEIDL, 2019).

Log de eventos real 8 (E8) e Log de eventos real (E9): Os logs de eventos são usados apenas para validar o tempo de processamento do algoritmo proposto em Maggi et al. (2013).

Log de eventos real 10 (E10): Em Yeshchenko et al. (2020a), são detectados dois *drifts* do tipo abrupto na perspectiva de fluxo em (05-25-2014), (09-24-2014) por meio do *Drift Chart* e um *drift* do tipo recorrente (não fica clara sua localização). Já em Yeshchenko et al. (2021) são encontrados *drifts* abruptos em 01-19-2011, 06-20-2011, 08-10-2012 e *drift* incrementais, recorrentes e graduais dentro de *clusters* gerados pelo algoritmo proposto. Esses *drifts* dentro dos *clusters* são relativos, pois a análise não é feita no log inteiro, sendo assim, é de validade questionável para a validação de outros algoritmos. Em Guan et al. (2023), encontrou-se dois *drifts* localizados nos traços 462 e 476. Segundo Guan et al. (2023), no traço 462 existe a introdução de novas relações de sequência direta, como "*IV Antibiotics*" $\succ$ "*ER Registration*", enquanto algumas relações anteriores desaparecem, como "*Admission NC*" $\succ$"*IV Antibiotics*". Da mesma forma, o *drift point* no traço 733 indica o surgimento de novas relações, como "*LacticAcid*" $\succ$"*Release B*", e a remoção de outras, como "*IV Liquid*" $\succ$"*ER Sepsis Triage*".

Log de eventos real 11 (E11): O log de eventos em questão provém do projeto CoSeLoG, em que foi realizado para examinar as semelhanças e diferenças entre vários processos de diferentes municípios na Holanda. O conjunto de dados é composto por cinco registros de eventos que anotam a execução de um processo de solicitação de licença de construção em cinco municípios anônimos distintos. As etiquetas/nomes das atividades nos diferentes logs de eventos se referem às mesmas atividades realizadas nos cinco municípios.

Guan et al. (2023) relatou que:

- *Local Complete-based Drift Detection* (LCDD), Lin et al. (2020), identifica seis pontos de mudança;
- TPCDD, Zheng, Wen e Wang (2017), encontrou quatro pontos de mudanças;
- AIMED, (Guan et al., 2023), só detectou dois pontos de mudanças.

Em Guan et al. (2023), afirma-se que no traço 202 há um *drift* onde o ramo contendo a transição, cuja etiqueta é "*Report reasons to hold request*", é um *gate* XOR que deixa de existir. No segundo *drift* (no traço 849), as atividades "*Determine necessity of stop advice*" e "*Determine necessity to stop indication*" não podem mais ser ignoradas.

Vale citar que em Huete, Qahtan e Hassani (2023), juntou-se os logs E4 e E8 para gerar um *drift* e validar o detector proposto *vis-à-vis* os existentes na literatura. No entanto, não delineou-se os detalhes sobre os *drifts.*

Observou-se que nos logs de eventos reais, em diferentes artigos, reportou-se *drifts* em localizações diferentes e de tipos diferentes. Isso revela a dificuldade de testar algoritmos de detecção de *drifts* com logs de eventos reais e, assim, esses logs de eventos não contém o *drifts* conhecidos e validados para que as ferramentas de detecção de *drift* sejam testadas e validadas.

Há vários logs de eventos citados na literatura que não estão disponíveis para a comunidade. Em Lu, Chen e Poon (2021a), Maaradji et al. (2017) e Ostovar, Leemans e Rosa (2020) e Omori et al. (2019) usaram-se logs de eventos reais não disponíveis para a comunidade.

Logs de eventos artificias com *process drift*

A revisão sistemática encontrou bases de logs artificiais com *process drifts*. Essas bases foram nomeadas como “**bases de logs de eventos artificiais existentes**”. As bases encontradas estão descritas abaixo e resumidas na (cf. Tabela 5), vale citar que na Tabela 5 utilizou-se o

termo **BLA** como abreviação para nomear essas bases de dados. Nesta tabela são listados em qual artigo a **base de logs de eventos artificias existente** foi apresentada, o número de log de eventos presente em cada log, a perspectiva e o tipo dos *drifts*, o número de traço dos logs, especificações em relação ao intervalo entre *drifts* e a existência de ruído. Essa tabela apresenta de forma geral as características das bases de logs de eventos artificiais existentes que foram subclassificação em sequência. A seguir foi feito a explanação de cada coluna da Tabela 5:

- Base artificial: identificador da "**base de logs de eventos artificias existentes**";
- Artigo: artigo em que a base é apresentada;
- Número de logs: número total de logs presente na base de dados;
- Perspectiva do *drift*: perspectiva em que os *drifts* foram inseridos nos logs de eventos, como trabalha-se com bases com conjuntos de logs de eventos pode ter logs com *drifts* em diferentes perspectivas;
- Tipos de *drifts*: tipos de *drift* existentes nos logs de eventos;
- Número de traços: número de traços de todos os logs de eventos;
- Intervalo entre os *drifts*: fixo, se os intervalos de traços entre os *drifts* dos logs de eventos são iguais, e variável, se os intervalos de traço entre os *drifts* mudam.

Tabela 5: Resumo das bases de logs de eventos artificiais com *process drifts* encontradas na literatura.

Base artificial	**Artigo**	**Nº de logs**	**Perspectiva do *drift***	**Tipos *drifts***	**Número de traços**	**Número de *drifts***	**Intervalo entre *drifts***	**Ruído**
BLA1	Adams et al. (2021)	1	Fluxo de e dados [14]	Abrupto	140	2	Fixo	Não disponibiliza
BLA2	Zellner et al. (2020)	4	Fluxo	Recorrente	1000	1	Variável	Não disponibiliza
BLA3	Maaradji et al. (2015)	72	Fluxo	Abrupto	2500,5000, 7500,10000	9	Fixo	Não disponibiliza
BLA4	Ceravolo et al. (2020)	942	Fluxo e temporal	Abrupto, recorrente, gradual, incremental	100,500, 1000	1	Fixo	Disponibiliza
BLA5	Ostovar, Leemans e Rosa (2020)	375	Fluxo	Abrupto	3000	2	Fixo	Disponibiliza
BLA6	Yang et al. (2022a)	34	"*Behavioural*" e Fluxo [15]	Abruptos e graduais	2000 e 2500	6 e 9	Fixo	Não disponibiliza

Fonte: O autor (2024)

[14] Existe um *drift* na estrutura do processo (fluxo) e um aumento na idade dos consumidores (dados).
[15] Existe logs de eventos com *drifts* na estrutura/fluxo do processo e log de eventos com "*behavioural drift*".

Base de logs de eventos artificiais existente 1 (BLA1[16]): Utiliza *CPN tools*[17] para gerar um modelo de reinvindicação de seguros, onde foi inserido um *drift* na perspectiva de dados com o aumento da idade dos consumidores e, consequentemente, um aumento nas notificações via *e-mail*. Adicionalmente, foi inserido um *drift* na perspectiva de fluxo no dia 133 e um segundo *drift* na perspectiva de dados no dia 132. O *drift* na perspectiva de fluxo é referente a mudança na frequência de relacionamentos diretos entre atividades sucessivas e o *drift* na perspectiva de dados é referente a distribuição da idade dos consumidores (ADAMS et al., 2021).

Base de log de eventos artificias existente 2 (BLA2[18]**)**: Os logs de eventos artificiais foram gerados com a ferramenta PLG2. O primeiro log de eventos começa com um processo composto por 6 atividades e duas portas XOR. Adicionalmente, foram criados mais três logs com 1.000 traços com *drifts*. Deve-se notar que foram inseridos *drifts* dividindo e intervalando partes dos desvios/portas com os *drifts*. O tipo de *drift* gerado foi o recorrente. Deve-se notar também que a distância entre os *drifts* foi alterada. Assim, os diferentes modelos de processos variam a cada 50, 100, 250, 500, 750 traços (ZELLNER et al., 2020).

Base de log de eventos artificias existente 3 (BLA3[19]**)**: Em Maaradji et al. (2015), disponibilizou-se uma base de dados com 72 logs criados a partir de diferentes parâmetros. A base de dados compreende um modelo de processo simples/didático referente à avaliação de pedidos de empréstimos como ponto de partida. Este modelo (cf. Figura 12), tem 15 atividades, incluindo estruturas de repetição, paralelas e caminhos alternativos (MAARADJI et al., 2015). A partir desse processo de origem foram geradas 12 variações do processo contendo *drifts* simples classificadas em três diferentes grupos: “I” (***I**nsertion*), “R” (***R**esequentialization*) e “O” (***O**ptionalization*). Para incluir *drifts* mais complexos foram combinados 3 *drifts* simples aleatoriamente gerando outros logs: “IRO”, ”IOR”, ”OIR”, “ORI”, “RIO”, ”ROI” (MAARADJI et al., 2015). As 12 mudanças simples estão listadas na Tabela 6.

[16] explainable_concept_drift_pm/pm4py/statistics/time_series/experiments/data at main · niklasadams/explainable_concept_drift_pm · GitHub
[17] http://cpntools.org/
[18] https://github.com/zellnerlu/DOA/tree/master/conjuntos de logs de eventos/synthetic
[19] https://data.4tu.nl/articles/conjunto de logs de eventos/Business_Process_Drift/12712436

Figura 12: Modelo de aplicação de empréstimos.

Fonte: Maaradji et al. (2015)

Tabela 6: Padrões de mudanças simples.

Código	Padrão de mudanças simples	Categoria
re	Adicionar/remover fragmento	I
cf	Tornar dois fragmentos condicionais/sequenciais	R
lp	Tornar fragmento repetível/não repetível	O
pl	Tornar dois fragmentos paralelos/sequenciais	R
cb	Tornar fragmento opcional/não opcional	O
cm	Mover fragmento para dentro/para fora de ramificação condicional	I
cd	Sincronizar dois fragmentos	R
cp	Duplicar fragmento	I
pm	Mover fragmento para dentro/para fora de ramificação paralela	I
rp	Substituir fragmento	I
sw	Trocar dois fragmentos	I
fr	Mudar frequência de ramificação	O

Fonte: Maaradji et al. (2015)

Dessa forma, foram gerados 18 novos modelos de processos indicando um *drift*, e para cada novo modelo foram gerados quatro logs contendo 2.500, 5.000, 7.500 e 10.000 traços. A ferramenta usada para esse fim foi o BIMP *Simulator*. Em cada log de eventos foram inseridos 9 *drifts* abruptos em intervalos de 10% do total de traços, substituindo o processo base pelo processo alterado, retornando ao processo base e repetindo o procedimento. Com isso, tem-se também um *drift* recorrente. Nos logs com 2.500 *traços*, os *drifts* ocorrem a cada 250 e assim os *change points* estão nos traços: 250, 500, 750, 1.000, 1.250, 1.500, 1.750, 2.000 e 2.250 (MAARADJI et al., 2015). Os logs de eventos presentes na **base de logs de eventos artificias existentes 3 (BLA3)** estão resumidos na Tabela 7. Porém, a base de dados alguns problemas como no número de traços e nos modelos em certos logs, esses logs foram identificados e expostos em Sato (2022).

Tabela 7: Resumo dos logs de eventos da BLA3.

Nº de Logs	**Log de Eventos**	**Número de *drifts***	**Perspectiva do *drift***	**Tipo do *drift***	**Número de traços**
18	Base x modelo alterado	9	Estrutural	Abrupto	2500
18	Base x modelo alterado	9	Estrutural	Abrupto	5000
18	Base x modelo alterado	9	Estrutural	Abrupto	7500
18	Base x modelo alterado	9	Estrutural	Abrupto	10000

Fonte: O autor (2024)

Assim, nessa base existem apenas logs de eventos com 2.500, 5.000, 7.500 e 10.000 traços, com intervalos fixos entre os *drifts* e *drifts* do tipo abrupto com recorrência. Contudo, tal parametrização foi utilizada em diversos artigos como base para testes: Liu, Huang e Cui (2018), Lu, Chen e Poon (2021a), Sousa et al. (2021), Seeliger, Nolle e Mühlhäuser (2017), Ostovar et al. (2016), Zheng, Wen e Wang (2017), Maaradji et al. (2017), Sato, Barddal e Scalabrin (2021), Lin et al. (2020), Adams et al. (2023b), Yang et al. (2022b), Hanga, Kovalchuk e Gaber (2022), Huete, Qahtan e Hassani (2023), Meira Neto et al. (2023), Meira Neto et al. (2024), Yaghoubi e Nazari (2021).

Vale citar que em Meira Neto et al. (2023), exclui-se os logs de eventos com 2500 traços, pois eles afirmam que distância reduzida entre os *drifts* e o tamanho da janela testada não permitem uma estabilidade entre os *drifts*. Em Gallego-Fontenla, Vidal e Lama (2023), replicou-se a geração dos logs de eventos, sem informar qual gerador foi usado, e classificou-se o padrão de mudança *fr* como "*behavioural drifts*". Além disso, em Sato (2022), algumas inconsistências nos logs de eventos foram encontrados e estão listadas na Tabela 8.

Tabela 8: Inconsistências encontradas na BLA3.

Nome do arquivo	Descrição
cb10k.xes	Contém apenas um drift após 5000 traços.
cd2.5k.xes	Contém 9 drifts após 250 traços; no entanto, o modelo alterado é completamente diferente do modelo base (também diferente da especificação BPMN).
cd7.5k.xes	Contém 9 drifts após 750 traços; no entanto, o modelo alterado é completamente diferente do modelo base (também diferente da especificação BPMN).
cm2.5k.xes	Contém 9 drifts após 250 traços; no entanto, o modelo alterado é o mesmo do cb em vez do definido na especificação BPM.
cm7.5k.xes	Contém 9 drifts após 750 traços; no entanto, o modelo alterado é o mesmo do cb em vez do definido na especificação BPM.
lp2.5k.xes	Contém 5000 traços com um drift após 500 traços.
lp5k.xes	Contém 10000 traços com um drift após 1000 traços.
lp7.5k.xes	Contém 15000 traços com drifts que não seguem a especificação; drifts em: 1000; 3500; 4000; 6500; 7000; 9500; 10000; 12500; 13000
lp10k.xes	Contém 15000 traços com drifts que não seguem a especificação; drifts em: 1000; 3500; 4000; 6500; 7000; 9500; 10000; 12500; 13000
re2.5k.xes	Contém 5000 traços com um drift após 500 traços.
re5k.xes	Contém 10000 traços com um drift após 1000 traços.
re7.5k.xes	Contém 15000 traços com drifts que não seguem a especificação; drifts em: 1000; 2000; 2500; 3500; 4000; 5000; 5500; 6500; 7000; 8000; 8500; 9500; 10000; 11000; 11500; 12500; 13000
re10k.xes	Contém 20000 traços com um drift após 2000 traços, e o log contém um loop (o mesmo dos logs lp) em vez de um re

Fonte: Adaptado de Sato (2022)

Base de logs de eventos artificias existentes 4 (BLA4[20]): Em Ceravolo et al. (2020) questiona-se alguns pontos das bases de logs artificiais gerados em Maaradji et al. (2015), em que os principais pontos são a falta (a) de ruídos nos logs, (b) diferentes tipos de *drifts*, e (c) de outras perspectivas como o tempo. Nela, são abordados os quatro tipos canônicos de *drifts* separados em perspectivas de fluxo e tempo. Em Ceravolo et al. (2020), usou-se o mesmo processo de negócio empregado em Maaradji et al. (2015), mas foram adicionadas transformações similares de acordo com a Tabela 9. Observou-se uma grande similaridade entre a base proposta de Ceravolo et al. (2020) e Maaradji et al. (2015). Essas bases foram geradas como *stream* de eventos. Uma *stream* de eventos armazena basicamente eventos com a informação do momento em que foram executados, permitindo uma análise *online*, sem a organização por *traços*. Porém, é necessária a identificação da instância do processo à qual cada evento pertence, permitindo assim que os métodos os agrupem em traços, se necessário.

[20] http://dx.doi.org/10.21227/2kxd-m509

Tabela 9: Padrões de mudanças simples utilizadas na BLA4.

Código	Padrões de mudanças simples	Categoria
cb	Tornar fragmento opcional/não opcional	O
cd	Sincronizar dois fragmentos	R
cf	Tornar dois fragmentos condicionais/sequenciais	R
cp	Duplicar fragmento	I
lp	Tornar fragmento repetível/não repetível	O
pl	Tornar dois fragmentos paralelos/sequenciais	R
pm	Mover fragmento para dentro/para fora de ramo paralelo	R
re	Adicionar/remover fragmento	I
rp	Substituir fragmento	I
sw	Trocar dois fragmentos	I

Fonte: Ceravolo et al. (2020)

Todas as *stream* de eventos geradas em Ceravolo et al. (2020) incluem um evento a cada 20 minutos. Adicionalmente, cada evento de um novo caso, que chega na *stream*, segue uma distribuição normal. Os *drifts* são inseridos sinteticamente com 100, 500 e 1000 casos e com ruído variável (5%, 10%, 15%, 20%). No caso do *drift* abrupto, o mesmo foi inserido na metade do log. Em relação ao *drift* recorrente foram inseridos *drifts* aproximadamente no primeiro e no segundo terço do log, considerando o número total de casos, tem-se que: para 100 casos, os *drift* estão nos casos 33 e 66; para 500, eles estão em 167 e 334; e para 1000, eles estão em 330, 660. No caso do *drift* gradual, a cada 20% da *stream* tem uma mudança de conceito. No *drift* incremental, 20% do *stream* foi dedicado ao modelo intermitente de mudanças, e a divisão foi 40-20-40. Na análise de *time*, a transição com 20% do *stream* foi subdivida em quatro partes com a diminuição de duração do intervalo entre eles. A Tabela 10 resume as informações dos logs de eventos da base.

Tabela 10: Resumo dos logs de eventos da BLA4.

Nº de Logs	**Log de Eventos**	**Nº de *drifts***	**Perspectiva do *drift***	**Tipo do *drift***	**Número de traços**	**Ruído (%)**
273	Base x modelo alterado	1	Estrutural	Abrupto	100, 500,1000	5,10,15,20
273	Base x modelo alterado	1	Estrutural	Recorrente	100, 500,1000	5,10,15,20
123	Base x modelo alterado	1	Estrutural	Incremental	100, 500,1000	5,10,15,20
273	Base x modelo alterado	1	Estrutural	Gradual	100, 500,1000	5,10,15,20

Fonte: O autor (2024)

Base de logs de eventos artificias existentes 5 (BLA5[21]): Em Ostovar, Leemans e Rosa (2020) gerou-se um conjunto de 375 logs artificias, utilizando o CPN *Tools*. O esquema consiste em um modelo estruturado em blocos com 42 atividades, com três estruturas diferentes, a saber: cinco XOR, seis AND e três *loop*. Para cada mudança de padrão, menos as duplicações de fragmentos, foram gerados cinco logs, cada um com dois *drifts* em fragmentos de tamanhos diferentes. Os logs foram gerados com 3.000 traços e com distância entre os *drifts* de 1.000 traços. Os *drifts* foram inseridos com uma mudança de padrão no primeiro *drift* (traço 1.000) e o segundo *drift* (traço 2.000) com reversão do processo ao processo original. Essa base foi utilizada para testar o algoritmo de detecção proposto em Meira Neto et al. (2024).

Base de logs de eventos artificias existentes 6 (BLA6[22]): Em Yang et al. (2022a), utilizou-se como conjunto de dados os logs de eventos com *drifts* repentinos de Maaradji et al. (2015) e criou-se 18 logs de eventos com *drifts* graduais e 16 logs de eventos com "*behavioural drift*" de forma repentina e gradual.

Para a geração dos 8 logs de eventos com "*behavioural drift*" repentinos, foram alteradas algumas probabilidades de transição no modelo base para o modelo alterado—ou com *drift* inserido artificialmente. Por exemplo, a probabilidade de transição das atividades foi ajustada de modo que, com maior frequência, os requerentes precisassem passar por uma avaliação mais rigorosa, tornando assim a aprovação da aplicação mais difícil. Em seguida, as instâncias de processo do modelo base original e dos registros recém-gerados foram combinadas para construir o log de eventos com *drift* (YANG et al., 2022a).

O eixo x dos gráficos da Figura 14 refere-se às 100 instâncias de processo denotados pelas caixas verde e laranja da Figura 13 (estrutura dos logs de eventos com *drifts* graduais). Enquanto que o eixo y liga à função de densidade de probabilidade que especifica os modelos A ou B das instâncias. As barras vermelhas e azuis são os modelos e as laranjas e verdes são os períodos de transição dos *drifts* graduais. Por exemplo, a primeira barra verde refere-se às instâncias de 200 a 300, onde o modelo de processo A é gradualmente substituído por B. A linha azul tracejada e a linha vermelha sólida representam as probabilidades de amostragem de instâncias do modelo A e B, respectivamente.

Esta estratégia (Figura 13) foi utilizada para gerar 18 logs de eventos com *drifts* graduais, utilizando os padrões de mudança de Maaradji et al. (2015), e 8 logs de eventos com

[21] https://drive.google.com/file/d/1xYuai8-HBrCZLSAuZMGPv7IJzOjbEsaY/view
[22] https://github.com/LingkaiYang/business-concept-drift

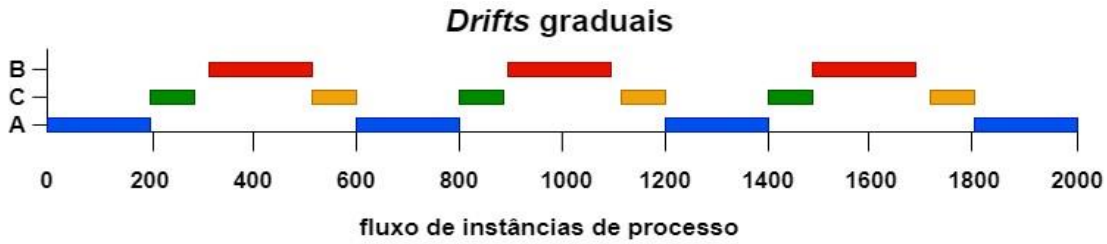
modelos que tem a alteração na frequência dos caminhos "*behavioural drift*" com transição gradual (YANG et al., 2022a). A Tabela 11 apresenta o resumo dos logs gerados na **base de logs de eventos artificias existentes 6**, nomeada como **BLA6**.

Tabela 11: Resumo dos logs de eventos da BLA6.

Nº de Logs	**Log de Eventos**	**Número de *drifts***	**Perspectiva do *drift***	**Tipo do *drift***	**Nº de traços**
8	Comportamento do Cliente (Abrupto)	9	*"behavioural" drift*	Abrupto	2500
18	Base vs. Alterado (Gradual)	6	Estrutural	Gradual	2000
8	Comportamento do Cliente (Gradual)	9	"*behavioural" drift*	Gradual	2500

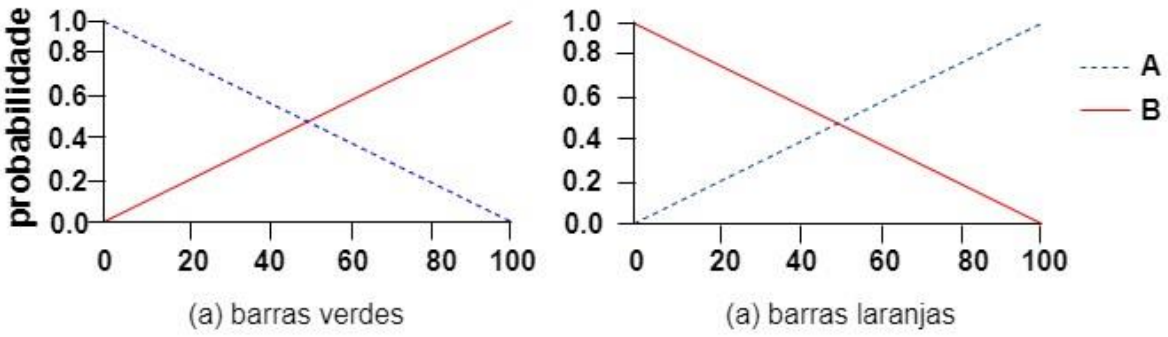
Fonte: O autor (2024)

Figura 13: Estratégia utilizada na geração de logs graduais.



Fonte: Yang et al. (2022a)

Figura 14: Probabilidade para "barras verdes" e "barras laranjas".



Fonte: Yang et al. (2022a)

Vale citar que em Sato et al. (2022) foi conduzida uma revisão sistemática da literatura com o objetivo de identificar as abordagens atuais para detectar *process drift* e nela as bases sintéticas — **BLA3**, **BLA4**, **BLA5** — são listadas no texto. Além disso, em Li et al. (2017) e Guan et al. (2023) foram apresentados detalhes sobre base de dados gerados com *drift*, sem disponibilizá-las.

Comparações dos algoritmos de detecção de *drifts*

Para comparar os algoritmos de detecção de *drifts* foram selecionados aqueles que se encontram disponíveis na forma de ferramentas computacionais e atendem a 3 critérios:

1. Reporta o ponto de detecção de mudança, ou seja, o ponto exato em que o *process drift* foi detectado;
2. Disponibiliza o código fonte;
3. Conduz a análise *offline*, ou seja, utiliza o log de eventos como entrada de dados.

As ferramentas que atenderam os critérios em questão foram: ProM-*Concept Drift plugin*, VDD *system*, Apromore-*ProDrift*, IPDD *framework*.

O ProM-*Concept Drift plugin* foi apresentado em Bose et al. (2011) e aprimorado com a análise de *drifts* graduais em Bose et al. (2014). A abordagem proposta aplica testes de hipóteses estatísticas em um *stream* de traços dividindo o log em *sub-logs*. O algoritmo compara dois *sub-logs* adjacentes e, caso exista uma diferença significativa entre os *sub-logs*, detecta um *drift*. Em Martjushev, Bose e Aalst (2015), incluiu-se um janelamento adaptativo em que o usuário não precisa fornecer o tamanho da janela (ou tamanho do *sub-log*).

Outra ferramenta é o Apromore-*ProDrift plugin* proposto em Maaradji et al. (2017) e Maaradji et al. (2015) que detecta *drifts* repentinos e graduais em *stream* de traços (abordagem runs) ou *stream* de eventos (abordagem eventos). Assim, na análise de logs de eventos (*runs approach*- abordagem *runs*) são aplicados testes de hipóteses em abstrações extraídas dos traços. A ferramenta faz a divisão de cada log em *sub-logs* para a comparação estatística com janelamento fixo e adaptativo, similar ao ProM-*Concept Drift plugin*. Os autores compararam suas estratégias, usando janelamento fixo e adaptativo para *drifts* abruptos, com o ProM-*Concept Drift plugin*. No entanto, não deixam explícito quais parâmetros foram usados na configuração do ProM-*Concept Drift plugin* para cada cenário e como foram calculados os componentes da métrica reportada, o *F-score*.

O VDD (*Visual Drift Detection*) *system*, Yeshchenko et al. (2021), é a ferramenta de detecção capaz de analisar os 4 tipos de *drifts* usando recursos visuais (*drifts maps, drift plots e autocorrelation plots*) e métricas (*Spread of Constraints, Erratic Measure, Augmented Dickey-Fuller Test*). Primeiramente, o VDD *system* divide o log em vários *sub-logs* para calcular a confiança usando restrições de tempo para cada sub-logs e para o log completo, resultando em séries temporais multivariadas; as limitações/condições temporais foram usadas

para permitir a detecção de mudanças de frequência e *drifts*. Em seguida, o VDD *system* agrupa as séries temporais multivariadas e aplica um algoritmo de detecção de ponto de mudança chamado PELT (*Pruned Exact Linear Time*). As métricas e as visualizações são geradas baseadas no agrupamento de restrições. Além disso, a avaliação do VDD *system* foi feita com a métrica *F-score*, mas o processo de cálculo da métrica não é claramente definido.

O IPDD[23] (*Interactive Process Drift Detection Framework*) é um *framework* projetado para aprimorar a detecção de *process drift*, abordando desafios comuns em ferramentas existentes, incluindo o usuário no processo de detecção e explicação da detecção, bem como expondo dificuldade de ajustar e comparar parâmetros dos detectores de *drifts*. Por meio de uma interface interativa. o IPDD facilita a experimentação com diferentes configurações de parâmetros, e oferece estratégias flexíveis de janelamento para análise de eventos (SATO; BARDDAL; SCALABRIN, 2021).

Inicialmente, o log de eventos analisado pelo IPDD é dividido em janelas com tamanhos fixos ou adaptativos, sobre os quais utiliza-se a descoberta de processo para obter modelos representativos para cada janela de análise. Posteriormente, esses modelos são comparados entre si, usando métricas de similaridade, para identificar diferenças que sinalizem *drifts*, permitindo a detecção dos *drifts* abruptos na estrutura do modelo.

Adicionalmente, foram identificados três artigos comparando as ferramentas de detecção de *process drifts*, são eles Ceravolo et al. (2020), Omori et al. (2019) e Adams et al. (2023b). Em Omori et al. (2019) comparou-se a acurácia do ProM-*Concept Drift Plugin vs*. Apromore-*ProDrift* usando uma base de dados real. No entanto, deve-se salientar que a comparação da acurácia usando uma métrica adequada (*F-score*) não pôde ser feita, uma vez que a localização dos *drifts* não é conhecida *a priori*. Já em Ceravolo et al. (2020), comparou-se a acurácia de cinco ferramentas de detecção de *process drifts* usando a **base de logs de eventos artificiais existentes 3** (**BLA3)** e a **base de logs de eventos artificiais existentes 4 (BLA4)**. Nessa comparação, combinou-se diferentes logs de eventos com diferentes parâmetros e os resultados foram analisados estatisticamente. Deve-se notar que eles utilizaram duas métricas tradicionais de regressão: MSE (*Mean Squared Error*) e RMSLE (*Root Mean Squared Logarithmic Error*). Essas métricas permitem comparar se o número total de *drifts* está próximo

[23] https://visual-pro-drift.com.br/

ao número real de *drifts*, mas não é possível avaliar qual a distância entre o real *process drift* e o *drift* detectado.

Em Adams et al. (2023b), foi proposto um *framework*[24] para comparar as ferramentas de detecção *de process drift*. Observou-se que as abordagens para detectar *process drifts* geralmente são avaliadas usando diferentes logs de eventos, dificultando a comparação direta entre as abordagens. Para permitir uma comparação mais generalizada, os logs utilizados em apenas uma única avaliação de artigo foram excluídos. Os logs restantes foram incluídos no *framework* caso atendessem dois critérios: disponibilidade pública e existência de um "*ground truth*" claramente definido para *process drifts*. Para garantir que a seleção dos logs de eventos fosse adequada para a avaliação pretendida, os autores estabeleceram critérios necessários para a escolha desses logs para cada objetivo de avaliação. Era crucial que os logs de eventos escolhidos contivessem um número significativo de *process drifts* verificáveis ("*ground-truth drifts*"), permitindo que os resultados da avaliação fossem validados com evidências estatísticas. Isso é necessário para avaliar a precisão, latência e sensibilidade aos parâmetros das ferramentas. A seleção final incluiu um total de 132 *drifts*, abarcando uma ampla variedade de tipos de *drifts* e um grande volume de dados, perfazendo 4.508.965 eventos (ADAMS et al., 2023b).

2.3.4 Geradores de logs de eventos

Na revisão da literatura realizada foram encontrados diversos geradores de logs de eventos, apresentados em ordem cronologia, a saber:

- Ratzer et al.(2003) apresenta o CPN tools[25];
- Burattin e Sperduti (2011) apresenta o PLG e, em sequência, Burattin (2016) apresenta o PLG2[26] (*Process Log Generator*);
- Stocker e Accorsi (2013). apresenta o SecSy;
- Pourmasoumi et al. (2015) apresenta uma ferramenta para geração de variantes implementada no PLG;
- Bolt, Leoni e Aalst (2016) apresenta o RapidProM;
- Yahya et al. (2016) apresenta o RT-PLG (*Real Time Process Log Generator*);

[24] https://github.com/cpitsch/cdrift-evaluation
[25] Gerador disponível em: https://cpntools.org/
[26] Gerador disponível em: http://plg.processmining.it e https://github.com/delas/plg

- Hmami, Sbai e Fredj (2021) apresenta o *Random Configurable Process Model Generator*;
- Herbet, Mangler e Rinderle-Ma (2021) apresenta o GENLOG[27];
- Grimm, Kraus e Aa (2022) apresenta a ferramenta **CDLG** (*Concept Drift Log Generator*);
- Grüger et al. (2023) apresenta o **DALG** (*Data Aware Event Log Generator*)[28].

Em resumo, a maioria desses geradores encerram como objetivo gerar bases de dados sintéticos de forma ampla, ou seja, bases que não contenham *drift* necessariamente, para a comunidade de mineração de processos e são brevemente abordados a seguir:

RT-PLG: Em Yahya et al. (2016) argumentou-se que as ferramentas para a simulação não guardam detalhes sobre o log de eventos, bem como não possibilitam quaisquer análise de dados. Contudo, há uma ferramenta que simula um processo e executa o log em tempo real, denominada de BP *Simulator*[29]. A geração de logs é dividida em cinco fases: preparação do modelo de processo, *parsing* do modelo de processo, processamento do log de evento em tempo real, descoberta do modelo de processo em tempo real e aprimoramento do modelo de processo. O RT-PLG usa CEP[30] (processamento complexo de eventos), uma tecnologia que permite que os eventos sejam processados continuamente antes de serem armazenados, para gerar eventos baseado no modelo de processo. Deve-se notar que o gerador demanda um modelo de processo como entrada e gera como saída um log de eventos em tempo real. Um log em tempo real guarda relações de dependência ao invés de eventos, ele é exibido em um visualizador de logs e armazenado/atualizado em tempo real. Assim, por definição do autor, um log em tempo real é uma coleção de dependências causais, cada dependência segue um traço, e não se baseia na sequência de atividades baseada no *timestamp.* Em Yahya et al. (2016), apresentou-se um log de eventos com 2.000 casos, 11.836 eventos e 16 atividades usando a ferramenta RT-PLG e validou-se os registros gerados com a ferramenta DISCO™ por meio da descoberta do modelo de processo.

PLG e PLG2: O PLG (*Process Logs Generator*) é uma ferramenta desenvolvida especificamente para criar *benchmark*s em *process mining*, tentando fazer face a escassez de

[27] Gerador disponível em: https://cpee.org/genlog/
[28] https://github.com/DavidJilg/DALG
[29] Gerador disponível em: https://www.bpsimulator.com/
[30] https://www.tibco.com/pt-br/reference-center/what-is-complex-event-processing

registros de processos de negócios reais e modelos correspondentes para avaliação de algoritmos de mineração de processos. O PLG permite ao usuário gerar um modelo de processo de negócio aleatório, mas realista, com base em parâmetros definidos pelo usuário, e "executar" esse processo para observar cada atividade. Isso é alcançado por meio de uma abordagem baseada na composição recursiva de padrões básicos conhecidos de processos, utilizando uma gramática específica para a geração desses modelos (BURATTIN; SPERDUTI, 2011).

Em seguida, Burattin (2016) apresenta o PLG2, evolução do PLG que foi anteriormente apresentado em Burattin e Sperduti (2011), capaz de gerar logs e *stream* de eventos. A ferramenta é capaz de simular log de eventos a partir de modelos de processos. O gerador foi desenvolvido em uma aplicação Java™, invocando também *scripts* Python™. Esse gerador possibilita que a base de dados seja gerada com várias versões do modelo (*control-flow drift*). O uso da ferramenta envolve as seguintes etapas: a importação de um modelo de processo, a geração de traços de processo aleatoriamente ou do enriquecimento do modelo existente para obter um modelo diferente. Deve-se notar que o PLG2 produz base de dados com ruídos. O autor cita que o ruído pode ser na organização do traço, no fluxo dos eventos e nos dados, porém, ele não detalha como o ruído foi inserido.

Em Burattin (2016), a geração de log artificiais de processos é baseada em padrões de fluxo de atividades: sucessão direta, execução paralela, sincronização de caminhos paralelos, execução mútua, convergência de caminhos e execução de subprocesso de forma repetida. Deve-se notar esses padrões foram combinados usando probabilidades, o que possibilita a criação de um modelo de processo completo. Tem-se a possibilidade de informar parâmetros para a inserção de ruídos e o número máximo de instâncias que ocorreram de forma paralela, e a escala de tempo para a criação de um *stream* contínuo. Além disso, o usuário tem a opção de definir as durações e os tempos das atividades. Para a geração de *drifts*, o PLG2 é capaz de alterar um modelo base ou o usuário deve fornecer o modelo base e o modelo alterado.

CPN *Tools*: Trata-se de uma ferramenta avançada para trabalhar com redes de Petri coloridas, permitindo a edição, simulação e análise de redes. Redes de Petri Coloridas (CPNs) são uma extensão das redes de Petri que incorporam dados, permitindo que cada token carregue um valor de um tipo específico. Elas modelam sistemas complexos descrevendo estados com lugares e ações com transições, utilizando expressões de arco para indicar mudanças de estado. Ao contrário das redes de Petri de baixo nível, as CPNs podem representar informações

detalhadas e variadas nos tokens, tornando-as mais expressivas para modelagem de sistemas (JENSEN, 1997).

A interface do CPN Tools utiliza "*binders*" e "*sheets*" para organizar e visualizar componentes das redes. Suporta múltiplas visões, manipulação direta e manual dos objetos, e oferece menus circulares e paletas de ferramentas para acesso rápido às funções. O CPN *Tools* realiza verificações de sintaxe e gera código de simulação automaticamente durante a edição, facilitando a análise e o teste de redes. Além disso, está preparada para expansões futuras, incluindo melhorias em simulação e recursos de animação. Em resumo, o CPN *Tools* combina uma interface com o usuário flexível, otimizando a modelagem e análise de CPNs (RATZER et al., 2003).

GENLOG: Em Herbet, Mangler e Rinderle-Ma (2021), tem-se como objetivo a criação de conjuntos de dados, *stream* de eventos de processos e séries de dados temporais em processos com poucos dados. O GENLOG é baseado em técnicas de aprendizagem de máquina e recebe como entrada um log no padrão XES. O gerador processa os logs de entrada e extrai diferentes séries temporais, uma para cada log de entrada. Vale citar que se deve informar o conjunto de medidas (atributos- cada atributo do log de eventos gera uma série de dados temporal) que serão extraídas de cada log de eventos. Em seguida, os dados são reagrupados em outra amostra para serem treinados em uma rede neural. O usuário deve decidir se as métricas serão agrupadas em lote (divisão do conjunto de dados em lotes menores) ou temporalmente (mantém a sequência temporal) para o treinamento de um modelo (rede neural). Dessa forma, os dados são gerados usando um método de predição com dados do treinamento. Por último, os dados são transformados em um log de eventos.

Em Herbet, Mangler e Rinderle-Ma (2021), usou-se uma **LSTM** (***L**ong **S**hort-**T**erm **M**emory*) para resolver o problema do desaparecimento de gradiente das redes neurais. A LSTM mantém um estado global, e usa *gates* para determinar se a informação deve ser mantida ou esquecida. O gerador possibilita que o treinamento seja contínuo usando os dados para treinar novos modelos. A técnica é indicada como adequada a processos que possuem *drifts*, uma vez que a LSTM está condicionada a dar ênfase nos dados mais recentes, considerando o log inteiro/completo. Os modelos de processos evoluem no tempo sobre o mesmo log. No entanto, uma das limitações da ferramenta é que pode ser necessário reiniciar o logs de eventos em caso de mudanças severas no modelo. Uma mudança na ordem de execução das atividades no modelo de processo é um exemplo.

RapidProM: Já em Bolt, Leoni e Aalst (2016), apresentou-se o RapidProM. Ele é um *framework*, disponível no ProM, que traz várias funcionalidades, dentre elas, a conversão de tipo do log de eventos. Deve-se notar que se trata de um *framework* que reúne ferramentas para o manuseio de log de eventos com a inserção de atividades diferentes, fusão de modelos, mudança de frequência das atividades. Sendo assim, esse *framework* pode ser utilizado como gerador de log de eventos sintéticos.

Random Configurable Process Model Generator : Em Baier, Reimold e Kuhl (2020), apresentou-se uma abordagem para a detecção de *drifts* e a sua implementação em um *toolset*, chamado "*random configurable process model generator*". Em Hmami, Sbai e Fredj (2021), esse *toolset* reúne um conjunto de funcionalidades, tais como: a geração de modelos de processos com variantes, a simulação de execuções e de geração de logs. Uma vantagem importante, em Baier, Reimold e Kuhl (2020) optou-se pela implementação das funcionalidades em questão no mesmo ambiente para facilitar o teste/avaliação da abordagem. O usuário pode escolher se o log de eventos tem ou não *drifts*, no caso do log com *drifts* deve-se escolher um dentre os quatro tipos de *drifts*: repentino, recorrente, gradual ou incremental. No entanto, não há nenhum parâmetro mais específicos para a inserção de cada *drift*, tal como uma função de decaimento no caso do *drift* gradual.

Em Hmami, Sbai e Fredj (2021), aprofundou-se nas funcionalidades e nos procedimentos de geração de log de eventos com a apresentação de um algoritmo de fusão e filtragem. Pode-se juntar logs de eventos com processos de negócios diferentes gerando um log com *drift*. A ferramenta foca na criação de novas variantes a partir de um modelo de processo. A partir de vários logs, a ferramenta filtra logs com parte similares, identifica os casos relevantes, junta todos os logs em um único log de eventos e filtra os eventos relacionados aos pontos de variação. Apenas três *gates* foram usados: AND, OR e pontos de variação. E o usuário tem as seguintes opções: gerar um modelo de processo configurável, especificando o número de pontos de variação e o número de variações para cada ponto; gerar variantes de processos; gerar, juntar e filtrar log de eventos.

PLG2 (gerador de variantes): Já em Pourmasoumi et al. (2015), apresentou-se diferentes algoritmos para a geração de pontos de mudanças. Três tipos de funções foram criados: a) funções de mudança de operadores, b) funções de mudança de atividades, e c) funções de mudança de conexão. As funções do primeiro tipo conseguem converter entradas AND para OR e OR para AND. Deve-se também informar: (a) o número de variantes, (b) o

índice de variação que especifica o grau de variação do modelo de processo de entrada, e (c) a função de distribuição de probabilidade (Gaussiana, Uniforme, Beta e Gama). Essa ferramenta foi implementada no PLG2 como gerador de variantes.

SecSy: Em Stocker e Accorsi (2013) argumentou-se sobre a necessidade de abordagens e ferramentas para a geração de logs sintéticos, em particular, para testar algoritmos de monitoramento e verificação de conformidade de processos. A ferramenta SecSy[31] conta com transformadores de modelos que são capazes de modificar os fluxos dos modelos: modificação de uma entrada AND para XOR, XOR para AND e troca da ordem de duas atividades consecutivas. A ferramenta é capaz de simular situações nas quais, em um intervalo de tempo, uma nova variante é registrada. O usuário pode, também, fornecer o modelo de processo e os dados dos atributos para gerar um log de eventos.

CDLG: Em Grimm, Kraus e Aa (2022), a ferramenta CDLG (*Concept Drift Log Generator*) foi desenvolvida para criar logs de eventos com *process drifts* no fluxo dos processos — repentinos, graduais, recorrentes e incrementais — permitindo aos usuários personalizar aspectos como complexidade do modelo, tamanho do log e presença de ruídos. Ela opera por meio de árvores específicas de processo para cada tipo de *drift,* gerando *sub-logs* que refletem os *drifts* desejados, e pode concatenar múltiplos *sub-logs* para simular um cenário com vários *drifts*. Além disso, a ferramenta oferece a opção de inserir ruído nos logs. Todas as informações cruciais sobre *drifts*, incluindo tipo, momento de ocorrência e mudanças de fluxo de controle, são registradas e armazenadas como atributos no arquivo XES resultante, facilitando a avaliação precisa das abordagens de detecção de *process drift* (GRIMM; KRAUS; AA, 2022). A validação da ferramenta foi feita no VDD *system*, no entanto, o artigo não apresenta uma validação estatísticas dos logs de eventos gerados pela ferramenta.

DALG: O DALG (*Data Aware Event Log Generator*)[32] apresentado em Grüger et al. (2023) é um gerador de logs de eventos de múltiplas perspectivas, que dá especial atenção à perspectiva de dados e à semântica associada. As variáveis são definidas por meio de um modelo acompanhado por um conjunto de informações semânticas, por exemplo, valores, dependências e distribuições. Ao combinar isso com um algoritmo de execução para criar a perspectiva de controle de fluxo, é possível gerar dados sintéticos que possuam valores de variáveis com significado semântico relevante. A geração da perspectiva de controle de fluxo

[31] Gerador disponível em: http://bit.ly/SecSy
[32] https://github.com/DavidJilg/DALG

se baseia em simulações que utilizam *tokens*, enquanto a parte dos dados segue um método baseada em regras (GRÜGER et al., 2023).

Em Stocker e Accorsi (2013) e Pourmasoumi et al. (2015), embora os autores não tenham focado em *process drift*, o método possibilita a criação de *drift*. O *drift* pode ser gerado colocando em prática as seguintes ações: gerar uma variante e a inserir no log de eventos. Logo, tem-se como resultado uma mudança no modelo de processo. E os geradores de variantes podem ser usados como uma ferramenta específica para gerar logs de eventos com *process drifts*.

A Tabela 12 mapeia os artigos que utilizaram os gerados supracitados na geração de logs de eventos.

Tabela 12: Geradores de logs de eventos sintéticos.

Gerador de logs	Citados em
CPN Tools	Huang et al (2018), Adams et al. (2021), Luengo e Sepúlveda (2012), Martjushev, Bose e Aalst (2015), Ostovar et al. (2017), Kumar, Thomas e Basava (2016), Ostovar et al. (2016), Bose et al. (2011), Burattin et al. (2015), Maggi et al. (2013), Che, Machu e Zhou (2016)
BIMP Simulator33	Liu, Huang e Cui (2018), Maaradji et al. (2017), Ceravolo et al. (2020), Maaradji et al. (2015), Xu, Zhang e Duan (2023), Sousa et al. (2024),
PIPE34	Zheng, Wen e Wang (2017), Yuan et al. (2020)
BPMN Modeler35	Ceravolo et al. (2020)
PLG ou PLG2	Hassani (2019), Zellner et al. (2020)
Random Configurable Process Model Generator	Hmani, Sbai e Fredj (2021)
PM4PY	Guan et al. (2023), Sousa et al. (2021),

Fonte: O autor (2024)

2.4 Considerações do Capítulo

A revisão sistemática, empreendida neste capítulo, permitiu identificar, *de um lado*, geradores de logs de eventos artificiais que facilitam, em uma certa medida, na inclusão de

[33] Gerador disponível em: http://bimp.cs.ut.ee/
[34] Gerador disponível em: https://sourceforge.net/projects/pipe2/.
[35] Gerador disponível em: https://demo.bpmn.io/

drifts em tais logs, e *de outro lado*, bases de dados sintéticas e reais contendo logs de eventos com *drifts*. Adicionalmente, identificou-se ferramentas capazes de modificar as variantes/caminho do processo de negócio, permitindo gerar os modelos e esses últimos podem ser passados como entrada para o gerador objeto desta dissertação, o **PDG** (*Process Drift Generator*).

Vale citar que foram encontrados geradores de logs de eventos com *drifts* na revisão sistemática que exigem a combinação de vários recursos para a geração de logs de eventos com *drifts* obedecendo critérios necessários da área de interesse, por exemplo, geração dos 4 tipos de *drifts*, controle do número de traços, intervalos livre entre os *drifts*, inserção de ruído no log de eventos. Adicionalmente, observou-se uma ausência clara de um protocolo de análise estatística para comparar métodos de detecção de *drifts*.

3 MÉTODO

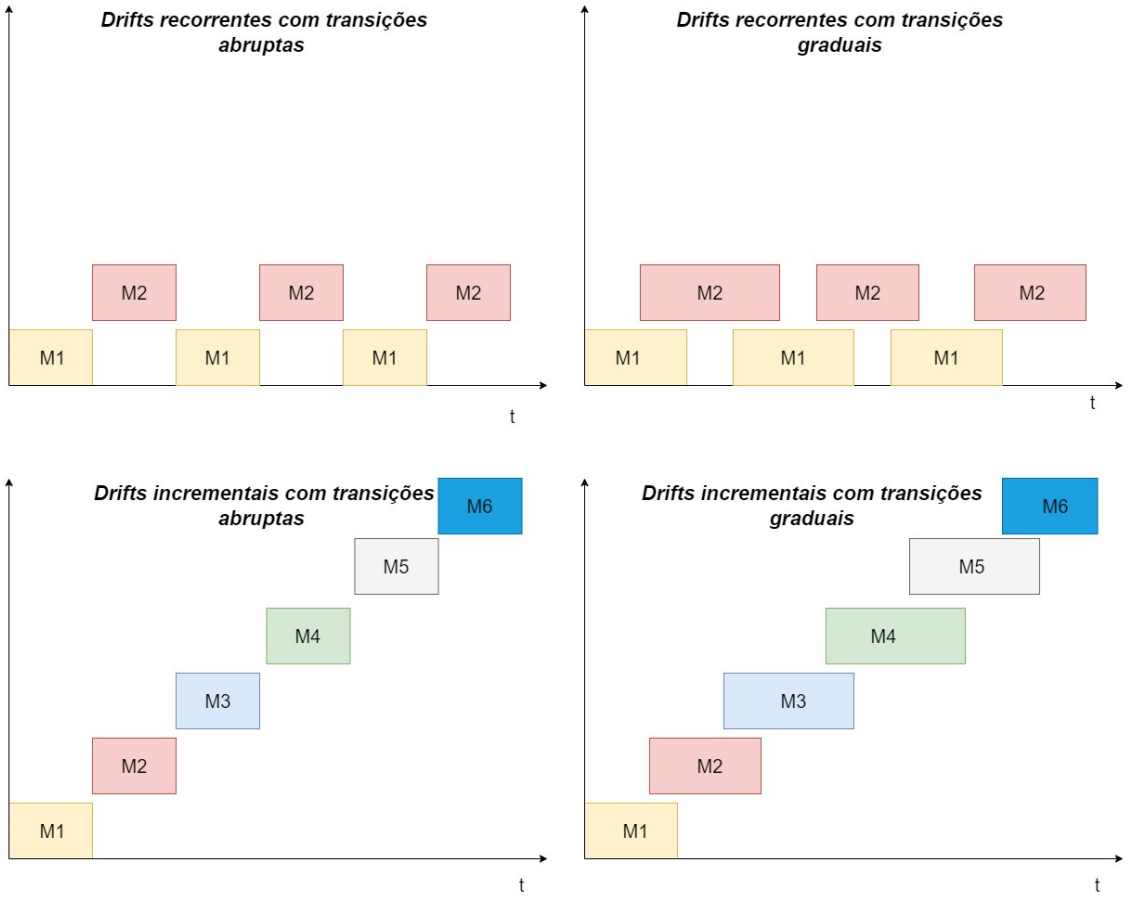
Esse capítulo apresenta a geração de logs de eventos com *drifts* e o protocolo de testes para comparação de desempenho de detectores de *drifts*. Em um primeiro momento são apresentadas as estratégias para geração dos logs de eventos no contexto do **PDG** (*Process Drift Generator*), Seção 3.1, e em um segundo momento são abordados os métodos colocados em prática para validação de logs de eventos com *drifts*, gerados conforme delineamento dado na Seção 3.2. Por último é detalhado o protocolo de análise estatística para comparação de ferramentas de detecção de *drifts*, cf. descrição na Seção 3.3.

3.1 Geração de logs de eventos com *process drifts* no fluxo do processo

Em um primeiro momento, o método foca na geração de logs de eventos com *drifts* estruturais, ou seja, com foco no fluxo do modelo de processo. Neste contexto, o **PDG**[36]. possibilita a geração de logs com *drifts* e para os seguintes tipos: abrupto, incremental, recorrente, gradual. Conforme a Figura 15, os *drifts* recorrentes e incrementais podem ter transições graduais ou abruptas entre modelos de processos. Em transições graduais, os modelos coexistem, enquanto em transições abruptas, não há coexistência. Os quatro algoritmos dessa seção, exibidos na sequência, foram classificados conforme a literatura e, para a geração de logs de eventos com *drifts* incrementais e recorrentes com transição gradual (cf. Figura 15), por exemplo, deve-se recorrer a função dedicada a "fusão" de logs. Adicionalmente, deve-se notar que implementação foi realizada para permitir a geração massiva, via linha de comando e os "*gold standard*" são inseridos no log conforme sugerido em Grimm, Kraus e Aa (2022) com a criação do atributo "*Real drifts*" no log de eventos.

[36] https://github.com/caioraduy/ProcessDriftGenerator.git

Figura 15: Transições para os tipos de *drifts* incrementais e recorrentes.



Fonte: O autor (2024)

3.1.1 Geração de logs de eventos com *drifts* abruptos

A geração de *drifts* abruptos denota-se por uma mudança repentina de um modelo para outro modelo. Neste contexto, a estratégia de geração de logs artificiais implementada parte de um conjunto de modelos M = $\{m_1, m_2, ..., m_k\}$, onde é gerado um conjunto de traços para um modelo $m_i \in$ M de forma que os traços de m_i não coexistam ao longo do log de eventos com os traços de quaisquer outros modelos $m_j \in$ M com $j \neq i$, gerando assim *drifts* abruptos. A estratégia básica para a geração de logs de eventos segue os seguintes passos (Algoritmo 1):

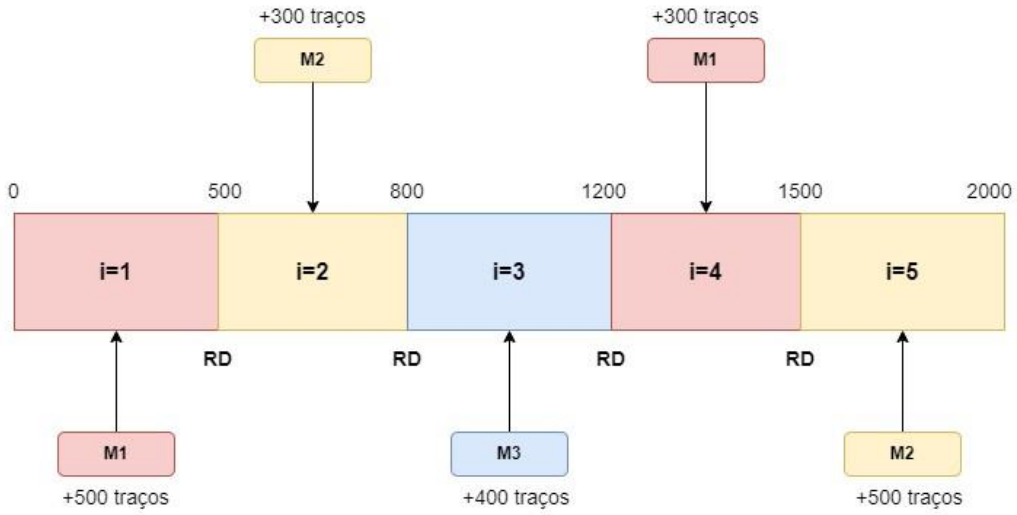
Algoritmo 1: Algoritmo simplificado para a geração de logs de eventos com *drifts* abruptos.

1. Obter conjunto de modelos de processos M = {m_1,m_2,...,m_k};
2. Obter conjunto de posições de inserção de *drifts* D = {d_1,d_2,...,d_{k-1}};
3. k ← |M|;
4. Calcular intervalo de traços T={<t_{a1},t_{b1}>,<t_{a2},t_{b2}>,..., <t_{ak},t_{bk}>}, a partir de D, onde ocorre apenas um modelo m_i;
5. q = |T|;
6. Criar log de eventos *L* vazio;
7. i ← 1;
8. Para cada intervalo de tempo t ∈ T faça:
    a. Se i ≥ k então i ← 1 senão i ← i + 1;
    b. Gerar x traços para m_i , *onde x é dado por* $t_{bi} - t_{ai}$
    c. Inserir o traço de m_i em *L*;
9. Exportar log de eventos *L* no formato XES.

Fonte: O autor (2024)

O esquema da Figura 16 ilustra o funcionamento a aplicação do método de geração. Neste esquema, toma como entrada um conjunto de posições de *drifts* D = {500, 800, 1200, 1500}, o número de traços N = 2.000, então calcula-se um conjunto de intervalos de traços T = {<0,500>, <501, 800>, <801, 1.200>, <1.201, 1.500>, <1.501, 2.000} em que cada modelo existe. Ao percorrer o conjunto T, os traços dos modelos dos processos m_1, m_2 e m_3 são adicionados no log de eventos L.

Figura 16: Geração de logs de eventos com *drifts* abruptos.



Fonte: O autor (2024)

3.1.2 Geração de logs de eventos com *drifts* graduais

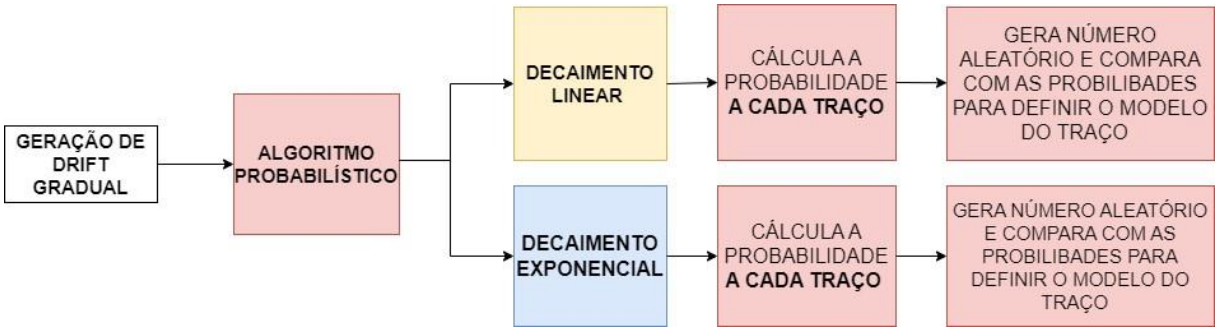
A geração de *drifts* graduais requer a definição de uma função de decaimento de probabilidade, conforme a Subseção 2.3.2, para determinar a coexistência de modelos no tempo. A especificação da função de decaimento é fundamental para que seja possível a geração do período de transição entre os modelos. Essa função é necessária para se ter uma probabilidade de comparação com qual modelo seguirá a geração simulada de traços. Nesta linha, definiu-se, na sequência, um algoritmo e suas funções para a geração de *drifts* graduais, sendo elas com os decaimentos *linear* e *exponencial*.

Neste contexto, serão detalhados, na sequência, os seguintes elementos:

1. funções de probabilidade com decaimento *linear* e *exponencial*;
2. cálculos de probabilidade por intervalo e por traço; e
3. algoritmo probabilístico.

A Figura 17 ilustra as funções envolvidas no algoritmo probabilístico, o qual está descrito no fim da Subseção.

Figura 17: Estrutura básica dos algoritmos para a geração de *drift* graduais.



Fonte: O autor (2024)

1) Função de probabilidade — decaimento linear

A estratégia para colocar em prática a geração de *drifts* graduais lineares consiste em fazer com que dois modelos coexistam parcialmente no tempo. Tal estratégia pode ser implementada da seguinte forma: dado dois modelos m_1 e m_2, onde m_1 é o modelo atual e ele coexiste com o modelo m_2 até que o modelo m_2 seja dominante no log de eventos. Se a transição de um modelo para outro é *linear*, então ela pode ser implementada por meio de uma função de

probabilidade—para a geração de *drifts*—descritas por duas retas. Uma reta crescente descreve a probabilidade de encontrar m_2 ao longo do intervalo do *drift* e uma decrescente descreve a probabilidade de encontrar m_1 ao longo do intervalo (cf. Figura 18).

A Figura 18 apresenta a evolução das probabilidades dos traços em cada log de eventos face a montagem dos modelos m_1 e m_2. Nesse exemplo, a inserção de um *drift* gradual começa no traço 1200 e termina no traço 1600. E o intervalo onde ocorre o *drift* tem comprimento de 400.

As equações (Equação 1, Equação 2, Equação 3 e Equação 4) a seguir fornecem os valores dos cálculos para a geração de traços de eventos descrevendo uma transição gradual para dois modelos de processos.

Equação 1: Intervalo do *drift.*

$$IntervaloDrift = tf - ti$$

Equação 2: Normalização do traço.

$$x = x - ti$$

Equação 3: Cálculo da probabilidade do modelo m_1.

$$P(m_1) = \frac{(intervaloDrift - x)}{intervaloDrift}$$

Equação 4: Cálculo da probabilidade do modelo m_2.

$$P(m_2) = 1 - P(m_1)$$

Onde,

*ti**:*** número de traços onde o *drift* começa;

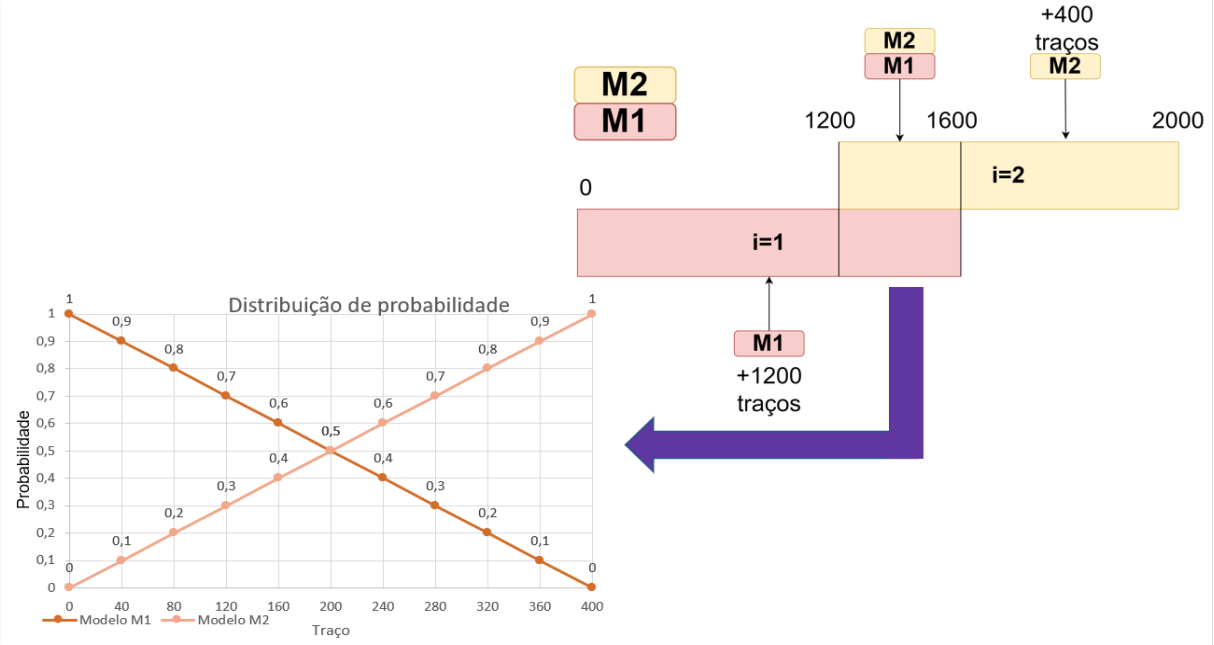
*tf**:*** número de traços onde o *drift* termina;

x: índice do traço em que a probabilidade está sendo calculada;

$P(m_1)$*:* probabilidade de existência do modelo m_1;

$P(m_2)$*:* probabilidade de existência do modelo m_2.

Figura 18: Probabilidade por modelo com decaimento linear ao longo do intervalo do *drift* gradual.



Fonte: O autor (2024)

2) Função de probabilidade – decaimento *exponencial*

A estratégia para colocar em prática a geração de *drifts* graduais exponenciais consiste em fazer, como no caso linear, que dois modelos coexistam sobre parte de suas existências. Tal estratégia foi implementada da seguinte forma: dado dois modelos m_1 e m_2, onde m_1 é modelo atual e ele coexiste com o próximo modelo m_2 até que o modelo m_2 seja dominante no log de eventos. Dado o mesmo intervalo de 400 traços, tem-se $\lambda = \frac{-\ln(0{,}01)}{400} = 0{,}011513$. A função exponencial não atinge $P(m_1) = 1$ devido a característica da função exponencial em que: $\lim_{x \to -\infty} e^x = 0$. Deve-se notar que a função de probabilidade para a geração de *drift* gera duas curvas assintóticas no intervalo em que foi especificado o *drift gradual* (cf. Figura 19). As equações (Equação 5, Equação 6 e Equação 7) são utilizadas para o cálculo do lambda e das probabilidades dos modelos.

Equação 5: Cálculo lambda da função probabilidade.

$$\lambda = -\frac{\ln(PL(m_1))}{IntervaloDrift}$$

Equação 6: Cálculo probabilidade do modelo m_1.

$$P(m_1) = e^{-\lambda.x}$$

Equação 7: Cálculo probabilidade do modelo m_2.

$$P(m_2) = 1 - P(m_1)$$

Onde,

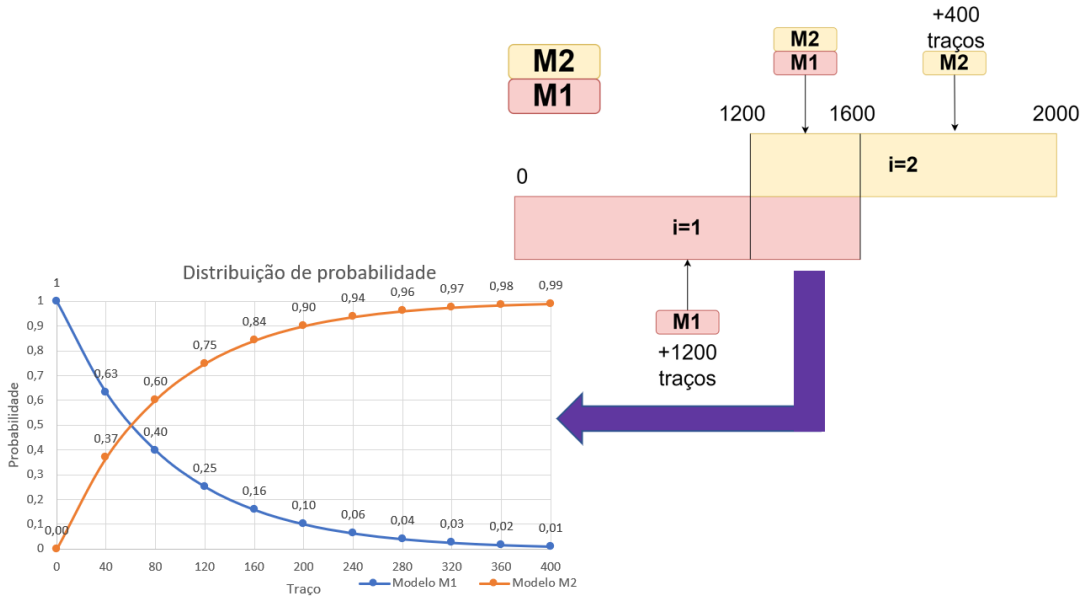
$P(m_1)$: probabilidade existência do modelo m_1;

$P(m_2)$: probabilidade existência do modelo m_2;

$PL(m_1)$: probabilidade do último traço de m_1.

$PL(m_1)$ dá a probabilidade do modelo m_1 existir. Ela deve ser maior que zero devido as propriedades da função logarítmica. Dessa forma, nos testes realizados utilizou-se 1%.

Figura 19: Probabilidade por modelo com decaimento exponencial ao longo do intervalo do *drift* gradual.



Fonte: O autor (2024)

3) Cálculo de probabilidade – por traço com decaimento *linear* ou *exponencial*

Quando se usa o cálculo de probabilidade por traço, calcula-se a probabilidade a cada traço e, a partir de uma dada função de geração de números aleatórios, gera-se um número aleatório para determinar se um dado traço será inserido no modelo m_1 ou modelo m_2. A probabilidade é calculada para cada traço para um dado intervalo de acordo com a função informada como parâmetro — decaimento *linear* ou *exponencial*. Em seguida, para cada traço uma função de geração de valores aleatórios gera um valor $x \in [0, 1]$. Se $0 \leq x \leq P(m_2)$, o traço é gerado a partir do modelo m_2 e se $P(m_2) < x \leq 1$, o traço é gerado a partir do modelo m_1. Matematicamente, esse cálculo pode ser dado pela Equação 8.

Equação 8: Determinação do traço.

$$S = \{0 \leq x \leq P(m_2) \rightarrow t = m_2 \vee P(m_2) < x \leq 1 \rightarrow t = m_1\}$$

Onde,

t: traço;

x: valor gerado aleatoriamente;

P(m₁): probabilidade existência do modelo m_1;

P(m₂): probabilidade existência do modelo m_2.

A função de geração de valores aleatórios em questão permite produzir logs de eventos similares, ou seja, os logs não iguais entre si e mais próximos da realidade, diferente de um método determinístico que se caracteriza por produzir logs de eventos bem mais controlados e previsíveis.

4) Algoritmo – probabilístico

O algoritmo para a geração de logs de eventos com *drifts* graduais se utiliza de dois modelos, que são: m_1 e m_2. Portanto, o algoritmo é capaz de gerar um log com *drifts* com transições graduais que são recorrentes, pois apenas dois modelos se alternam. E um intervalo em que o *drift* gradual ocorre; esse intervalo é informado. Neste contexto, o log de eventos começa com traços do modelo m_1, e a partir do primeiro traço em que o *drift* aparece, começa-se a adição de traços de m_2. Nesse intervalo em que o *drift* ocorre: traços de m_1 e m_2 são adicionados seguindo uma dada função de probabilidades (e.g., *linear* ou *exponencial*). O cálculo de probabilidades é feito por traço e utiliza um gerador de números aleatórios para decidir de qual modelo o traço será gerado.

O algoritmo foi definido com base na seguinte lógica: o log começa com traços do modelo atual m_1 e na sequência ele recebe traços do modelo m_2. Após o *drift* gradual, m_2 se torna o modelo corrente. Essa alternância é feita *n* vezes; deve-se notar que o número de *drifts* é parâmetro informado por meio da posição dos *drifts*. A estratégia em questão segue os passos dado no Algoritmo 2.

Algoritmo 2: Algoritmo simplificado para a geração de *drifts* graduais.

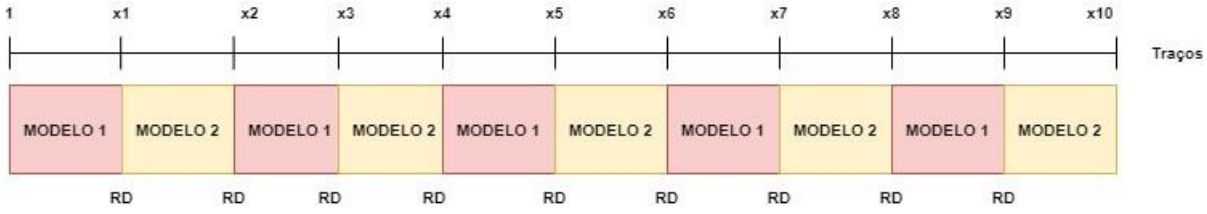
1. Criar log de eventos vazio L;
2. Obter os modelos de processos m_1 e m_2;
3. Obter intervalo de *drift* D = $<t_{b1}, t_{a2}>$;
4. Definir m_1 como o modelo atual e m_2 como o próximo modelo;
5. Gerar traços para o modelo atual até que comece o *drift* e adicionar traços gerados no log L;
6. Faça para cada traço do subintervalo d ∈ D:
    a. se o intervalo tem um *drift*
        i. Calcular a probabilidade de existência de traços dos modelos m_1 e m_2 de acordo com a função de probabilidade informada;
        ii. Gerar número aleatório x entre 0 e 1:
            1. se $0 \leq x \leq P(m_2)$, adicionar em L traço do modelo m_2;
            2. se $P(m_2) < x \leq 1$, adicionar em L traço do modelo m_1;
    b. se o intervalo não tem *drift*
        i. gera traços do modelo atual
        ii. redefine o modelo atual como modelo próximo
7. Exportar log de eventos *L* no formato XES.

Fonte: O autor (2024)

3.1.3 Geração de logs de eventos com *drifts* recorrentes

A geração de *drifts* recorrentes consiste na justaposição de dois modelos que se alternam. Seja M = {m_1, m_2} um conjunto de modelos. Os *drifts* são posicionados nos traços: $x_1, x_2, x_3, ..., x_{n-1}$, onde n - 1 é o número de *drifts*—ou quantidade de vezes que os modelos se alternam. O esquema da Figura 20 ilustra a geração de um log de eventos com 9 *drifts* recorrentes.

Figura 20: Esquema de geração de log de eventos com *drifts* recorrentes.



Fonte: O autor (2024)

Deve-se notar que cada *drift* é inserido na marcação RD (*Real Drift*), sob o esquema de sequenciamento dos processos.

Estratégia de geração de log de eventos com *drifts* recorrentes opera basicamente com a mesma lógica do *drift* abrupto, entretanto ela envolve apenas dois modelos de processos que se alternam n vezes. Os seguintes passos geram um *drift* recorrente com transições abruptas conforme dado pelo Algoritmo 3.

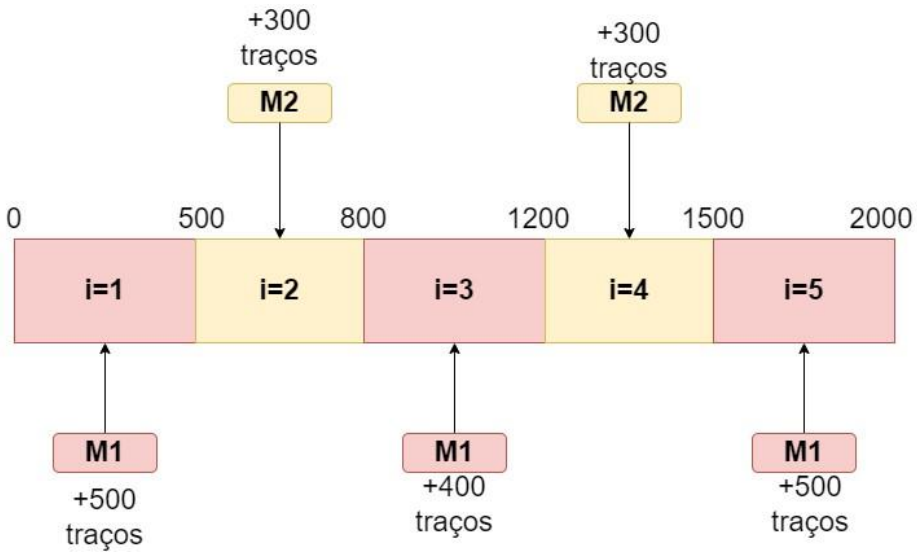
Algoritmo 3: Algoritmo simplificado para a geração de logs de eventos com *drifts* recorrentes.

1. Obter conjunto de modelos de processos M = {m_1,m_2,...,m_k};
2. Obter conjunto de posições de inserção de *drifts* D = {d_1,d_2,...,d_{k-1}};
3. k ← |M|;
4. Calcular intervalo de traços T={<t_{a1},t_{b1}>,<t_{a2},t_{b2}>,..., <t_{ak},t_{bk}>}, a partir de D, onde ocorre apenas um modelo m_i;
5. Criar log de eventos *L* vazio;
6. i ← 1;
7. Para cada intervalo de traços <t_i,t_f> ∈ T faça:
   Se i ≥ k então i ← 1 senão i ← i + 1;
   Faça j de t_i até t_f:
   Gerar traço r a partir do modelo m_i e adicionar r em L;
8. Exportar log de eventos *L* no formato XES;

Fonte: O autor (2024)

Uma simplificação do algoritmo em questão, seria assumir que há apenas dois modelos m_1 e m_2 que se alternam. Deste modo, quando o índice do modelo *i* for ímpar gera-se traços de m_2 e quando *i* for par gera-se traços de m_1. Em outras palavras, a estratégia de geração *drifts* recorrentes consiste na alternância dos modelos m_1 e m_2 de forma abrupta. Para tal, o algoritmo—que implementa essa estratégia—gera traços com base em modelos diferentes. Por exemplo, dada um conjunto de posições de *drifts* D = {500, 800, 1200, 1500} e número de traços t = 2.000. O algoritmo calcula um conjunto de traços T = {<0,500>, <501, 800>, <801, 1.200>, <1.201, 1.500>, <1.501, 2.000>}, percorre-se o conjunto D, iniciando com intervalo i = <0, 500>, gera-se traços do modelo m_1 e adiciona-os em L. Deve-se notar que cada *drift* é inserido na marcação RD (*Real Drift*), sob o esquema de sequenciamento dos processos da Figura 21.

Figura 21: Geração de logs de eventos com *drifts* recorrentes.



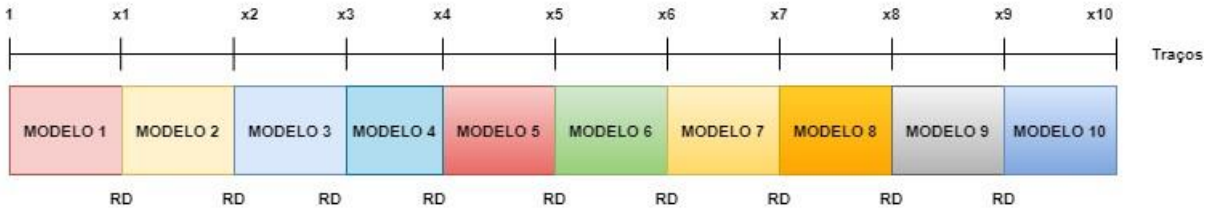
Fonte: O autor (2024)

Deve-se notar que cada *drift* é inserido na marcação RD (*Real Drift*), sob o esquema de sequenciamento dos processos.

3.1.4 Geração de log de eventos com *drifts* incrementais

Um log de eventos com *drifts* incrementais contém *n* modelos de processos com pequenas alterações crescentes a partir do modelo base. Seja M = {$m_1, m_2, m_3, ..., m_n$}, geram-se (*n-1*) *drifts* posicionados nos traços: $x_1, x_2, x_3, ..., x_{n-1}$. O esquema da Figura 22 ilustra a geração de *drifts* incrementais com 10 modelos e 9 *drifts*. Vale ressaltar que para que ocorra um *drift* do tipo incremental, as variações dos modelos devem referir-se ao mesmo modelo base.

Figura 22: Exemplo de log de eventos com *drifts* incrementais.



Fonte: O autor (2024)

Analogamente aos esquemas anteriores, os *drifts* são inseridos nas marcações RD ("*Real Drift*"). E a estratégia para a geração de log de eventos com *drifts* incrementais parte do princípio de que, cada *drift* é gerado por um novo modelo com alterações em relação ao modelo

anterior. Tem-se então que o número de modelos informados deve ser igual o número de pontos de *drifts* mais um. A geração de *drifts* incrementais implementa uma lógica envolvendo uma estrutura de repetição que percorre um conjunto de intervalos de traços T e gera logs de eventos na ordem em que os modelos são informados no conjunto de modelos M.

O Algoritmo 4 a seguir descreve a lógica básica do gerador de *drifts* incrementais.

Algoritmo 4: Geração de logs de eventos com *drifts* incrementais.

```
1. Obter conjunto de modelos de processos M = {m1,m2,...,mk};
2. Obter conjunto de posições de inserção de drifts D = {d1,d2,...,dk-1};
3. k ← |M|;
4. Calcular intervalo de traços T={<ta1,tb1>,<ta2,tb2>,..., <tak,tbk>}, a
   partir de D, onde ocorre apenas um modelo mi;
5. Criar log de eventos L vazio;
6. i ← 1;
7. Para cada intervalo de traços <ti,tf> ∈ T faça:
            i ← i + 1;
            Faça j de ti até tf:
                 Gerar traço r a partir do modelo mi e adicionar r em L;
8. Exportar log de eventos L no formato XES;
```

Fonte: O autor (2024)

3.1.5 Inserção de ruído no log de eventos

A inserção de ruído foi implementada retirando o primeiro evento de um dado traço *j*. A estratégia consiste em dividir o número total de traços com ruído por 10 e retirar o primeiro evento de cada intervalo. O primeiro eventos foi escolhido para que fosse possível fazer a verificação/validação da retirada dos eventos. O usuário deve informar a porcentagem de traços com *drift* quando ele especifica o log que ele deseja gerar. Por exemplo, para o seguinte objetivo: gerar um log de eventos com 1.000 traços e 5% de ruído. O log é divido em um conjunto intervalos I = {<1,100>, <101,200>, <201,300>, <301,400>, <401,500>, <501,600>, <601,700>, <701,800>, <801, 900>, <901,1000>} com | I | = 10. O log de eventos terá 50 traços com ruído, 5 traços por intervalos. E os traços com ruídos ($n \in$ N) estarão localizados nos intervalos dos primeiros 5 traços: N = {<1,5>, <101,105>, <201,205>, <301,305>,

<401,405>, <501,505>, <601,605>, <701,705>, <801, 805>, <901,905>}. Deve-se notar que cada intervalo $n \in N$ é um subintervalo de um intervalo $i \in I$ (cf. Algoritmo 5).

Algoritmo 5: Algoritmo simplificado para a inserção de ruído no log de eventos.

```
1. txRuído ← 0.05
2. a ← txRuído * |L|
3. offset ← a / 10
4. Para i de 1 até |L| passo |L| / 10 faça:
        idx ← i + offset
        Para j de i até idx faça:
           Remover o primeiro evento do traço j
5. Exportar L no formato XES.
```

Fonte: O autor (2024)

3.1.6 Fusão de logs de eventos com *drifts*

Para a geração de logs de eventos com mais de um tipo de *drift*, desenvolveu-se uma estratégia que permite mesclar diversos logs de eventos em apenas um. Antes de mesclá-los, precisa-se ajustar o *timestamp* e o *case id* dos eventos para manter consistente o log de eventos final. Com esse simples procedimento, tem-se a possibilidade de fazer diferentes combinações de *drifts*, caso seja necessário. A estratégia segue os seguintes passos (cf. Algoritmo 6):

Algoritmo 6: Fusão de logs de eventos.

```
1. Dado um conjunto de logs de eventos L = {l1, l2, ..., ln}
2. Dado um log de eventos E vazio.
3. Dado um identificador ID ← 0
4. Para cada log q ∈ L faça:
      Para cada traço t ∈ q faça:
           ID ← ID + 1
           t.attributes['Concept:name'] ← ID
           Para cada evento ev ∈ t faça:
                ev['time:timestamp'] ← += 0.005
      Adicionar traço modificado t no log de eventos E
5. Exportar E no formato XES
```

Fonte: O autor (2024)

3.1.7 Gerador de traços

A geração de traços foi implementada tomando como base o comportamento do método "*basic playout"*, disponível no PM4PY[37]. Deve-se salientar que foram feitas algumas modificações após a utilização do gerador de eventos do PM4PY, foram adicionados os atributos "*lifecycle:transition*", "*org:resource*" e "*resource*", os quais receberam valores nulos. Essas modificações foram necessárias para tornar possível testar os logs de eventos gerados na ferramenta Apromore *ProDrift* (selecionada para comparação) que requerem a existência dos atributos mesmo que estejam vazios. Deve-se notar também que os intervalos de tempo entre os eventos gerados por padrão no PM4PY são fixos e iguais a 1s. Esse valor foi modificado para 20s, mas pode ser alterado como parâmetro de entrada do gerador. Sugere-se a utilização de valores fixos para que não sejam criados possíveis *drifts* na perspectiva temporal nos logs de eventos. Os logs gerados seguem o formato canônico XES.

Tal modificação permite escolher o *timestamp* de entrada. Por padrão, o PM4PY gera traços com *timestamp* iniciais idênticos e, quando eles são juntados, os *timestamp* ficam repetidos. Logo, os traços se sobrepõem e não há nenhuma alteração no modelo. Essa alteração foi feita ao longo do código implementado.

A geração de logs de eventos com *drifts* abruptos, graduais, recorrentes e incrementais foram descritos nesta seção, assim como, os diferentes comportamentos para a geração de traços. Deve-se frisar que as estratégias em questão colocaram em prática comportamentos lineares e exponenciais.

A segunda parte do projeto concerne a validação dos logs produzidos.

3.2 Validação dos logs de eventos

A seguir são detalhadas as estratégias utilizadas para a validação dos logs de eventos gerados pelo **PDG** e, para isso, a métrica de avaliação *F-score* será abordada.

3.2.1 Métricas de avaliação

A métrica *F-score* (Equação 11 e Equação 12) é conhecida na área da mineração de dados e representa a média harmónica entra a precisão e o *recall*. Ela é calculada com base nos

[37] https://pm4py.fit.fraunhofer.de/documentation

valores de TP (*True Positive*), FP (*False Positive*), e FN (*False negative*). Na análise de *process drifts*, um TP indica um *drift* detectado que ocorreu no processo, um FP indica um *drift* detectado que não ocorreu e um FN é um *drift* real que não foi detectado pela ferramenta (MARTJUSHEV; BOSE; AALST, 2015). A seguir são detalhadas as estratégias para o cálculo do *F-score*, bem como o protocolo de análise estatística dos resultados.

Em alguns cenários, a métrica *precision* (Equação 9) é mais importante do que a métrica *recall* (Equação 10) e vice-versa. Por exemplo, em um problema de detecção de pacientes com câncer, o *recall* é mais importante porque a classificação errada (falsos negativos, ou seja, diagnosticar erroneamente um paciente com câncer como não tendo câncer) é inaceitável. Por outro lado, a *precision* é mais importante para cenários em que estamos apenas interessados em falsos positivos. No contexto de detecção de *process drifts*, é insuficiente detectar tanto falsos positivos (considerar *drifts* que não aconteceram como *drift*) quanto falsos negativos (não considerar *drifts* reais como *drifts*). Como resultado, utiliza-se o *F-score* que é a média harmônica entre a *precision* e o *recall*, como medida de desempenho adequada (YANG et al., 2022a).

Equação 9: *Precision.*

$$precision = \frac{TP}{TP + FP}$$

Equação 10: *Recall.*

$$recall = \frac{TP}{TP + FN}$$

Equação 11: *F-score.*

$$F - score = 2\frac{precision \times recall}{(precision + recall)}$$

Equação 12: *F-score* em termos de TP, FP e FN.

$$F - score = \frac{TP}{(TP + \frac{FP + FN}{2})}$$

Onde,

TP: *True Positive*;

FP: *False Positive;*

FN: *False Negative.*

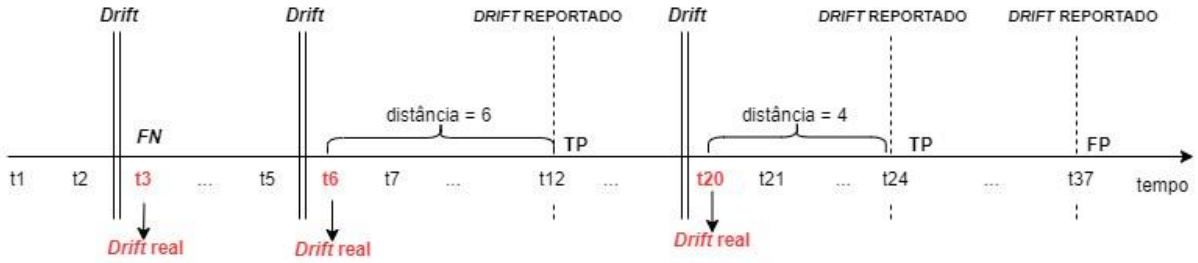
A seguir são detalhadas as estratégias para o cálculo do *F-score* e, bem como a apresentação da estratégia de cálculo escolhida.

F-score* e *mean delay

Na investigação conduzida por Sato (2022), definiu-se um intervalo para os TPs, onde RD representa a marcação do *Real Drift* e et o número de traços até o próximo *drift* conhecido. Se o log de eventos apresentar apenas um *drift*, um TP é registrado quando a mudança é identificada após o ponto da mudança real. O *F-score* varia de 0 a 1, avaliando a eficácia na detecção de mudanças. No entanto, essa pontuação não avalia a proximidade entre o ponto de mudança identificado e o ponto de mudança real. O cálculo do "*mean delay*" complementa a análise do *F-score*, quantificando o intervalo entre os *drifts* detectados e os reais para pontos considerados como TPs.

A Figura 13 mostra um exemplo de aplicação dessas métricas. No eixo x, as marcas são organizadas por *timestamp* (t_1, t_2, ..., t_n). O log de eventos possui três *drifts* reais nos pontos: t_3, t_6 e t_{20}. O método de detecção aponta três *drifts*, no exemplo: t_{12}, t_{24} e t_{37}. O *F-score* é considerado como TP somente quando um *drift* é detectado após um *drift* real e antes do próximo *drift* conhecido. No exemplo, somente t_{12} e t_{24} são reconhecidos como TP. A distância entre *o drift* detectado e o real é o *mean delay*.

Figura 23: Cálculo do *F-score* e do *mean delay.*



$$F - score = \frac{TP}{TP + \frac{(FP + FN)}{2}} = \frac{2}{2 + \frac{(1+1)}{2}} = 0{,}67$$

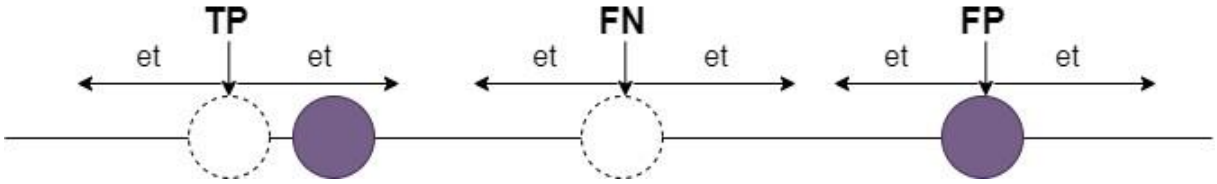
$$Mean\ delay = \frac{\sum_{i=0}^{TP} distância(\ drift\ reporado, drift\ real)}{TP} = \frac{6+4}{2} = 5$$

Fonte: Adaptado de Sato (2022).

F-score considerando erro de tolerância (reativo e preditivo)

Em Yang et al. (2022a), afirma-se que *F-score* com erro de tolerância é adequado para *drifts* repentinos. No entanto, em *drifts* graduais, os *reals drifts* (RDs) estão localizadas em um período contínuo em vez de um ponto específico. Portanto, um ponto de mudança detectado é considerado um TP se estiver localizado em qualquer uma das regiões de mudança reais (YANG et al., 2022a). No entanto, em Yang et al. (2022a), o intervalo considerado [RD - *et*, RD + *et*], englobam *drifts* que foram detectados antes mesmo de acontecer como TPs. A Figura 24 representa como os TPs, FNs e FPs são computados, quando se considera o erro de tolerância antes e depois do *drift* real, sendo o *drift* real o círculo em pontilhado e o *drift* detectado o círculo roxo.

Figura 24: *F-score* com erro de tolerância reativo e preditivo.

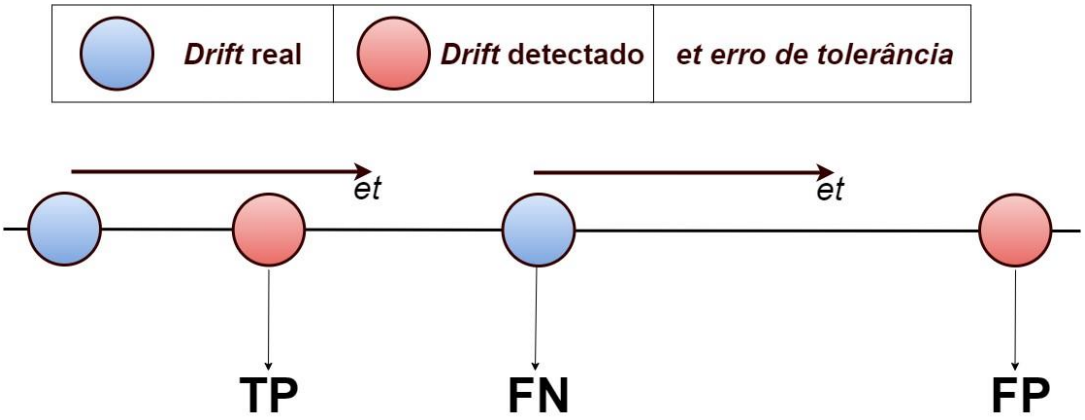


Fonte: Modificado de Martjushev, Bose e Aalst (2015)

F-score considerando erro de tolerância (reativo)

Os mecanismos de detecção são reativos; ou seja, eles precisam dos eventos após o *drift* real para detectar uma mudança. É essencial contar apenas um TP se a posição do *drift* foi reportada após a posição do *real drift* (RD) e considerar uma janela de tolerância. Utilizou-se também o parâmetro *et* (*error tolerance*), proposto em Zheng, Wen e Wang (2017). Logo, apenas um TP é contabilizado se o *drift* ocorreu no intervalo [RD, RD + *et*]. A Figura 25 exemplifica as três classificações para o cálculo do *F-score*: (a) TP – o *drift* detectado está dentro do intervalo aceitável; (b) FN – o *drift* real não foi apontando pela ferramenta no intervalo aceitável; e (c) FP – o *drift* apontado está fora do intervalo aceitável (RADUY et al., 2023).

Figura 25: *F-score* com erro de tolerância reativo.



Fonte: O autor (2024)

O *F-score* considerando erro de tolerância (reativo) foi utilizado em todos os experimentos realizados ao longo do documento, pois ele considera o erro de tolerância apenas

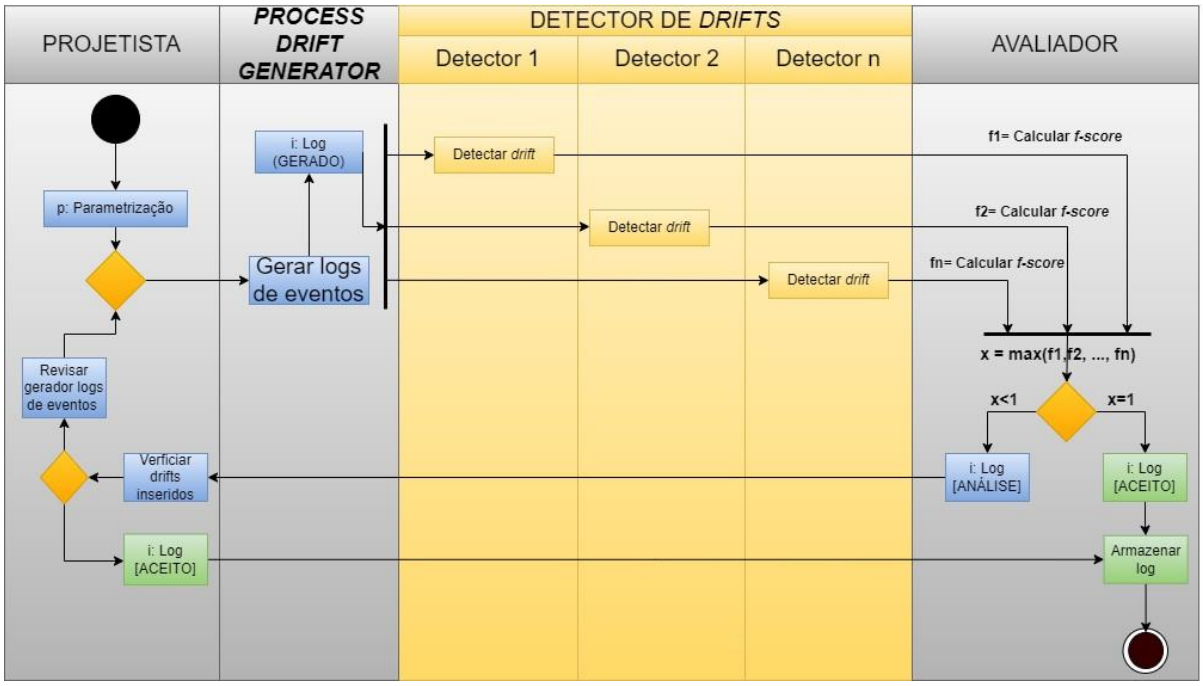
após a ocorrência do *drift* e mede a acurácia com apenas uma métrica, ou seja, não é necessário analisar o *mean delay* em conjunto, o que torna mais fácil a visualização gráfica dos resultados.

3.2.2 Validação de logs com *drifts* com transições abruptas

O processo de validação utilizado para a aceitação de logs de eventos com transições abruptas segue o passo a passo da Figura 26:

1. utilizar o **PDG** para geração dos logs para a composição da "**Base Gerada 2**";
2. submeter cada log de eventos a pelo menos 2 detectores de *drift* da literatura;
3. calcular o *F-score*;
4. se pelo menos um dos detectores, dada uma configuração testada, encontrou um *F-score* igual a 1, assume-se que os *drifts* foram inseridos corretamente;
5. se nenhum dos detectores, dada uma configuração testada, encontrou um *F-score* igual a 1, não se pode assumir que os *drifts* foram inseridos corretamente. Portanto, o log de eventos deve ser analisado em profundidade/manualmente (por ex. verificação do arquivo XES, descoberta do modelo, análise dos traços, etc.);
6. se na revisão manual, são encontrados os *drifts* no log, assume-se que o log de eventos está correto, mesmo que o detector de *drift* não tenha detectado os *drifts*;
7. se na revisão manual, não são encontrados os *drifts*, o log está com problemas e a estratégia do **PDG** deve ser investigada/corrigida.

Figura 26: Processo de validação da "**base gerada 2**".



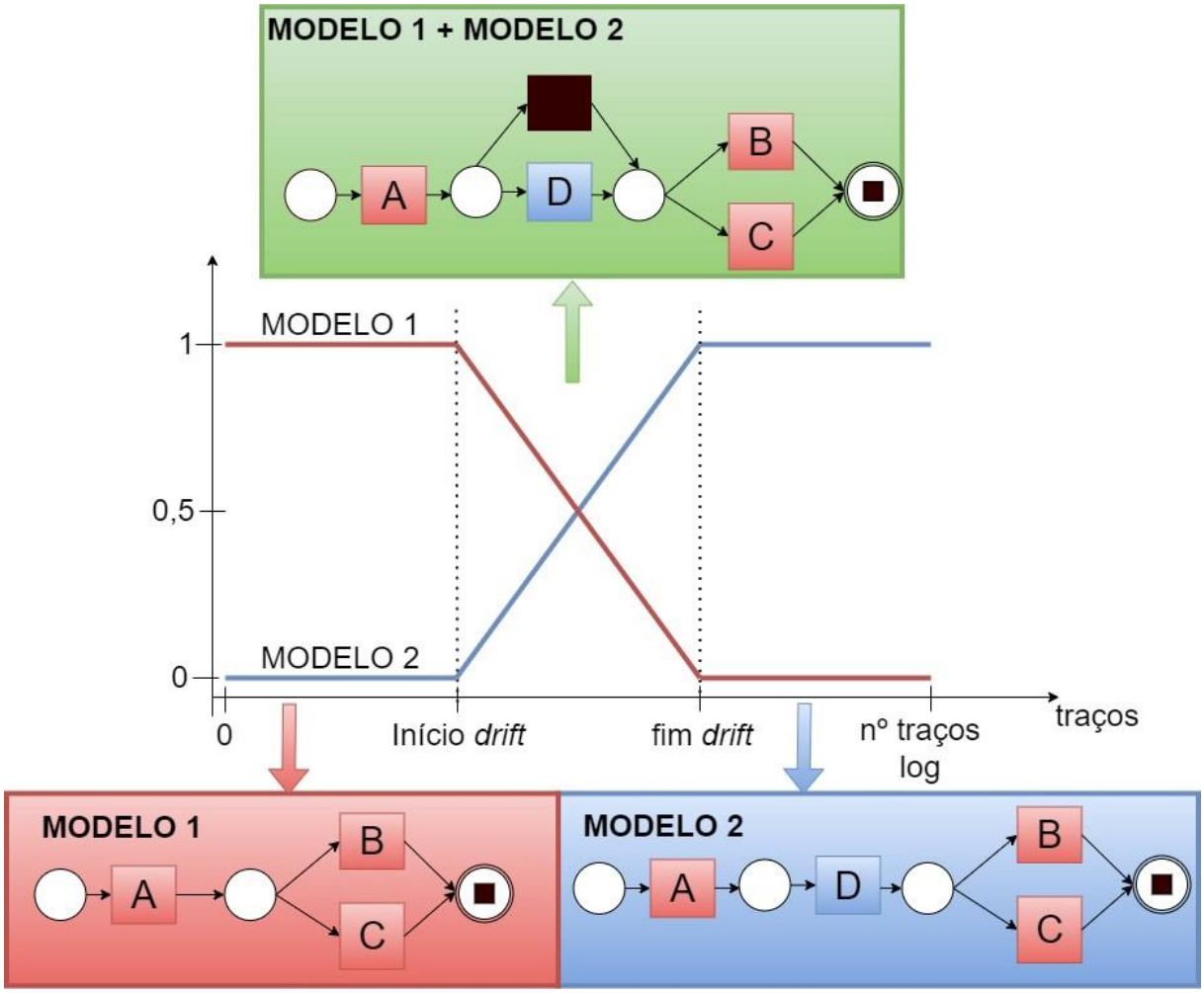
Fonte: O autor (2024)

Adicionalmente, para validar a evolução dos modelos conforme *drifts* incrementais ou recorrentes deve-se recorrer a extração do modelo por meio da divisão do log em *sub-logs* e comparação visual.

3.2.3 Validação de logs com *drifts* com transições graduais

O processo de validação dos logs com *drifts* com transições graduais consiste em explorar o log de eventos extraindo os modelos por intervalo de traços e analisa-los visualmente. Durante período de transição gradual, deve-se obter um modelo que combine traços dos dois modelos. Tomando como exemplo, tem-se a Figura 27 em que durante período de transição (início do *drift* até o final do *drift*), o modelo gerado combina o Modelo 1 com o Modelo 2. No caso em específico (cf. Figura 27), o Modelo 2 é o Modelo 1 como uma atividade adicionada e, dessa forma, no período de transição terão traços que vão pular a atividade adicionada e traços que vão passar por ela gerando um Modelo 3 (Modelo 1 + Modelo 2). Deste modo, com a análise visual dos modelos gerados pelos algoritmos mineradores pode-se concluir se os *drifts* estão realmente presentes no log.

Figura 27: Processo de validação do log de eventos com *drifts* graduais.



Fonte: O autor (2024)

Outra estratégia para verificação aproximada da corretude do processo de geração de logs de eventos com drifts é avaliar a capacidade do gerador em replicar um modelo existente.

3.2.4 Comparações das réplicas com base de logs artificias da literatura

O processo de validação dos logs de eventos gerados (cf. Figura 28) consiste em reproduzir logs de eventos e comparar de forma estatística. Este processo consiste em quatro etapas:

1. gerar *n* amostras, tomando como base a mesma parametrização (tamanho dos logs de eventos, tipo de *drifts*, posição do *drifts*) de uma base de dados existente na literatura;
2. aplicar o processo de validação de logs com *drifts* com transição abrupta;
3. aplicar os detectores de *drifts* em ambas as bases de dados;
4. comparar os resultados obtidos pelos detectores de *drifts* sobre cada log de eventos da base de dados original *versos* cada log de eventos de cada amostra da base de dados da réplica gerada pelo **PDG**.

Em outras palavras, os logs de eventos são gerados pelo **PDG** com os mesmos parâmetros da base de logs da literatura ou original, em seguida aplica-se o método de validação de logs com transições abruptas (cf. Subseção 3.2.2), submetendo-os as ferramentas de detecção de *drifts*. Os logs de eventos que não apresentam *F-score* igual a 1, em pelo menos uma configuração, são investigados por meio de um processo de divisão do log em *sub-logs*, extração do modelo e avaliação visual. Em seguida, a **base original** e a **base gerada (**reproduzida**)** devem ser submetidas as ferramentas de detecção de *drift* com parâmetros/configurações iguais. E em seguida, pode-se obter os *F-scores* para cada log de **base original** e a **base gerada (**reproduzida**)**. As amostras dos logs de eventos replicados são comparadas utilizando o teste de *Wilcoxon* e se todos os resultados são significativamente iguais assume-se que os logs são iguais, caso contrário, deve ser feita a inspeção do log de eventos (cf. Figura 28).

O Algoritmo 7, a seguir, mostra a dinâmica envolvendo as detecções de *drifts* e a comparação dos *drifts* detectados *log-a-log*, considerando ambas as bases de dados: **base original** e **base gerada (**reproduzida**)**.

Algoritmo 7: Dinâmica da validação dos logs de eventos gerados.

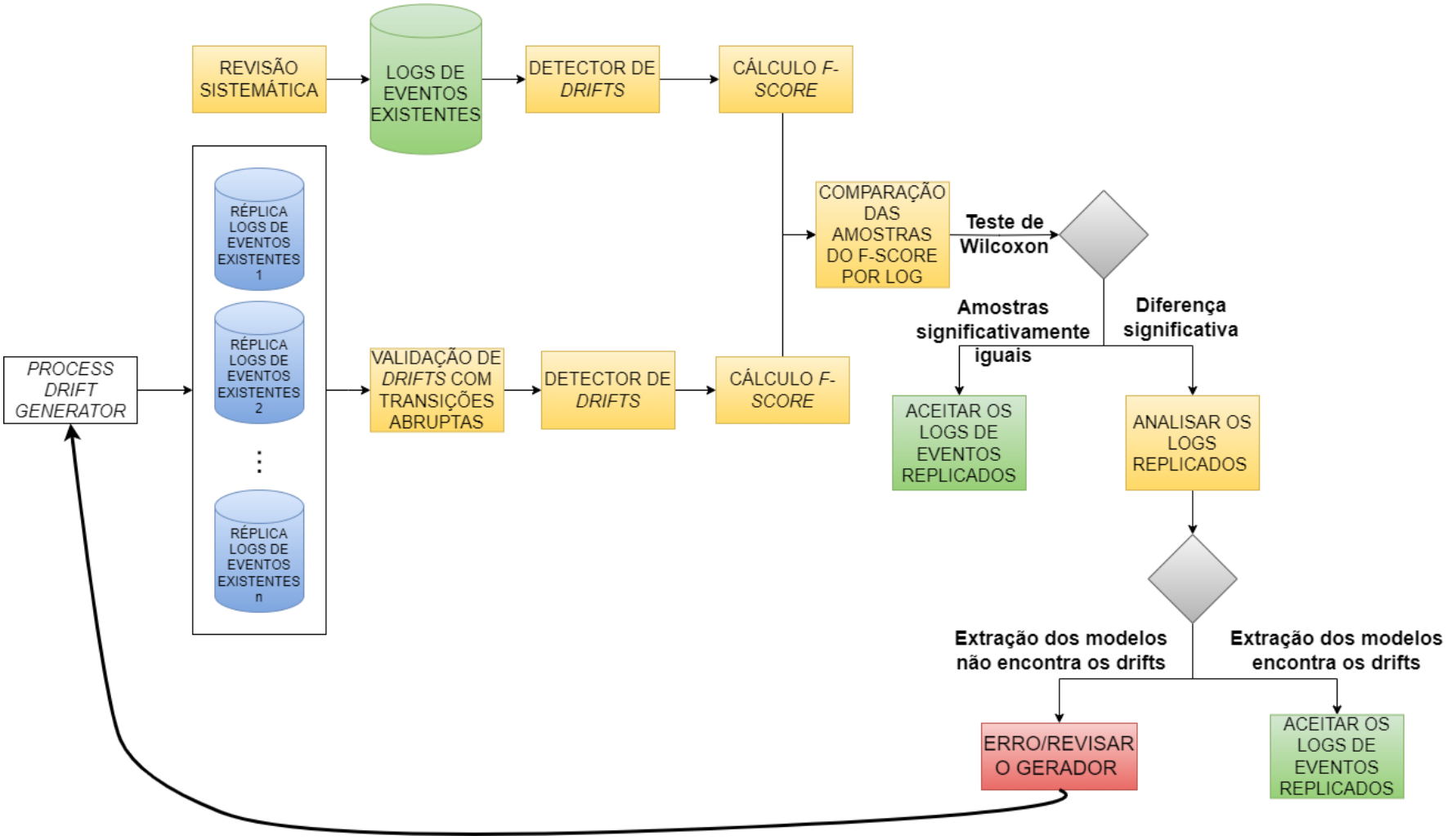
```
C ={}
para cada log i ∈ base original:
      di = detectar drifts(i)
      fi = calcular F-score(i, di)
      para cada log j ∈ base réplica {i}:
            dj = detectar drifts(j)
            fj = calcular F-score(j, dj)
            C = C ∪ aplicar Wilcoxon(fi, fj)
```

Fonte: O autor (2024)

Deve-se salientar, que assumiu-se aqui a premissa de que se a mesma ferramenta obtiver os mesmos *F-scores* para os dois logs (ex. log original da **base existente** e log gerado da **base gerada (**reproduzida**)**) com um grau de significância de 5%, os dois logs são estatisticamente iguais.

A seguir, detalha-se o teste de *Wilcoxon*.

Figura 28: Fluxo para a validação dos *logs* de eventos reproduzidos.



Fonte: O autor (2024)

Teste de *Wilcoxon*

O teste de *Wilcoxon* é baseado no escore-diferença ($d_i = X_i - Y_i$), sendo X e Y duas amostras com o mesmo número de elementos e condições pareadas. A hipótese nula é quando as amostras X e Y são equivalentes, ou seja, tem a mesma mediana e a mesma distribuição contínua. As equações básicas são Equação 13, Equação 14, Equação 15, Equação 16, conforme definido em Siegel e Castellan Jr (2006):

Equação 13: Média

$$\mu_{T^+} = \frac{N(N+1)}{4}$$

Equação 14: Variância

$$\sigma^2_{T^+} = \frac{N(N+1)(2N+1)}{24}$$

Equação 15: Variância com o decréscimo da variabilidade de T

$$\sigma^2_{T^+} = \frac{N(N+1)(2N+1)}{24} - \frac{\sum_{j=1}^{g} t_j(t_j-1)(t_j+1)}{2}$$

Equação 16: Valor z

$$z = \frac{T^+ - \mu_{T^+}}{\sigma_{T^+}}$$

Onde,

N: número de pares combinados subtraindo o número de pares em que os elementos são iguais (di=0);

T^+: soma dos postos dos d_i's positivos;

T^-: soma dos postos dos d_i's negativos;

z: valor z;

μ: média;

g: número de grupos de postos empatados diferentes;

tj: número de postos empatados no grupo j;

σ^2: variância;

σ: desvio padrão.

Após a definição das equações é possível enumerar as atividades para a aplicação do teste de *Wilcoxon:*

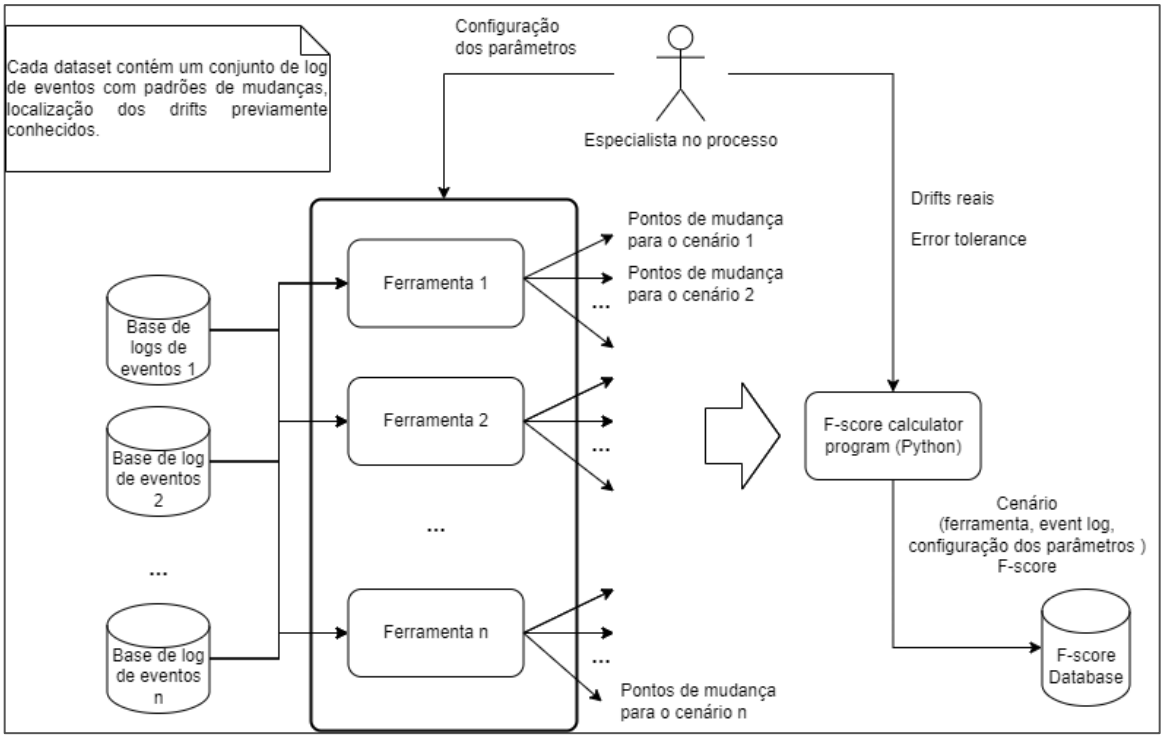
1. Determinar a diferença com sinal $d_i = X_i – Y_i$ entre as duas variáveis, para cada par combinado X_i e Yi.
2. Atribuir postos a estes d_i's sem considerar os sinais. Para di's empatados, atribuir a média dos postos empatados.
3. Afixar em cada posto o sinal (+ ou –) do d que ele representa.
4. Determinar N, o número de d_i's não-nulos.
5. Determinar T^+, a soma dos postos que têm um sinal positivo.
6. Determinar a significância do valor observado de T^+ depende do tamanho de N:
    a. Se N é 15 ou menos
        i. comparar a probabilidade associada de T^+ conforme os valores críticos de T^+ para os testes de postos com sinal de *Wilcoxon*.
        ii. Se a probabilidade associada com o valor observado de T^+ for **menor ou igual ao nível de significância escolhido, deve-se rejeitar a H_0.**
    b. Se N é maior do que 15
        i. Se não existem postos empatados, calcular o valor de z usando as equações: Equação 13, Equação 14 e Equação 16;
        ii. Se existem postos empatados, corrigir e calcular o valor z usando as equações: Equação 13, Equação 15 e Equação 16;
        iii. Determinar a probabilidade associada consultando as probabilidades associadas com a cauda superior distribuição normal. Para um teste bilateral, duplicar o valor da probabilidade dada. **Se a probabilidade assim obtida é menor ou igual a α, rejeitar H_0.**

3.3 Protocolo para a comparação de ferramentas de detecção de *process drifts*

A definição do protocolo experimental para a comparação de detectores de *process drifts* foi feita com base na estatística para tornar a tarefa confiável—independente do observador. Tal protocolo experimental segue o esquema geral exibido da Figura 29, que compara a acurácia de cada ferramenta de detecção de *drift* para saber qual ferramenta tem melhor resultado na detecção correta dos *drifts*. Os resultados obtidos no *pipeline*—os pontos

de *drifts*—são usados para o cálculo do *F-score* considerando erro de tolerância (reativo) e armazenados em uma base de dados, que em um segundo passo são agrupados por ferramenta de detecção para comparação (RADUY et al., 2023).

Figura 29: Protocolo experimental para a comparação das ferramentas de detecção de *process drifts* estruturais com transições abruptas.



Fonte: Raduy et al. (2023)

Para a comparação das ferramentas de detecção de *drifts* estruturais com transições abruptas, detalhada no Capítulo 5, foram selecionadas quatro ferramentas: Apromore-*ProDrift*, ProM-*Concept Drift*, VDD *System* e IPDD *Framework*. E as abordagens avaliadas foram com janelamento adaptativo e fixo:

- Apromore-*ProDrift*: janelamento adaptativo e fixo
- ProM-*Concept Drift*: janelamento adaptativo
- VDD *system*: janelamento fixo
- IPDD *framework*: janelamento adaptativo e fixo

A aplicação do protocolo de testes para a comparação dos detectores de *drift* foi feita com duas **bases de logs de eventos artificiais existentes** e uma **base gerada pelo PDG**. O fluxo de avaliação de cada ferramenta segue o esquema da Figura 29 associando-a a uma base de dados e um conjunto de parâmetros para os cenários a serem analisados.

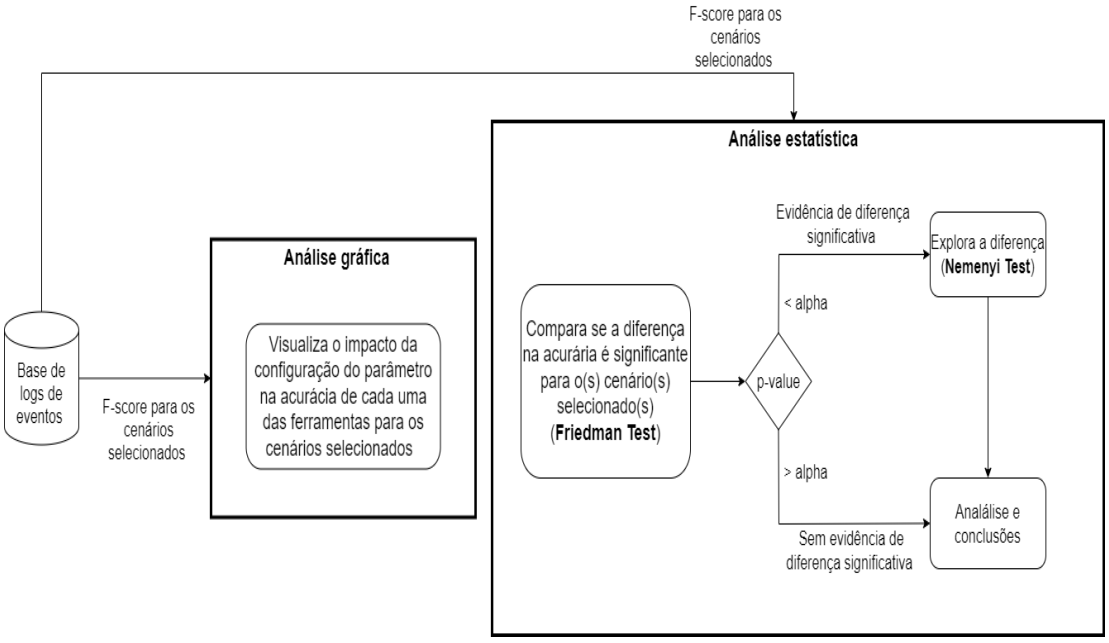
Foram feitas três análises/comparações:

(a) a influência do tamanho da janela nas abordagens com janelamento fixo;

(b) a influência do tamanho da janela inicial nas abordagens com janelamento adaptativo; e

(c) a acurácia entre todos os métodos considerados.

Figura 30: Análises para o *F-score* calculado.



Fonte: Raduy et al. (2023)

O cálculo do *F-score* foi sistematizado com uma rotina de cálculo computacional para possibilitar um volume elevado de experimentos. Além do cálculo do *F-score*, foram também propostos dois tipos de análise, a saber: gráfica e estatística (cf. Figura 30). Na análise estatística, o conjunto de dados é submetido ao Teste de *Friedman* para verificar se há evidência diferença significativa entre as amostras e, caso existam, aplica-se o Teste de *Nemenyi.*[38] As amostras são pareadas e submetidas aos mesmos cenários (RADUY et al., 2023). Os testes estatísticos aplicados no protocolo são descritos nas próximas subseções.

3.3.1 Teste de *Friedman*

O teste de *Friedman* para análise de variância de dois fatores por postos avalia a hipótese *nula* de que *k* medidas repetidas ou grupos combinados originam-se da mesma população ou de

[38] https://colab.research.google.com/drive/1OQgBBvTo3LDuBqwC3CalTmQnT_l8GCkH?usp=sharing

populações com medianas idênticas. Para definir a hipótese *nula* de forma mais precisa, considera-se θ_j como a mediana da população na *j*-ésima condição ou grupo. Deste modo, a hipótese *nula*, que postula a igualdade das medianas, é formulada como H_0: $\theta_1 = \theta_2 = ... = \theta_k$. A hipótese alternativa, por sua vez, é apresentada como H_1: $\theta_i \neq \theta_j$ para ao menos um par de condições ou grupos i e j, significando que, caso a hipótese alternativa seja verdadeira, existe pelo menos um par de condições com medianas distintas (SIEGEL; CASTELLAN JR, 2006).

No teste de *Friedman*, organiza-se os dados em uma tabela com *N* linhas e *k* colunas, onde as linhas simbolizam os elementos das amostras combinados, e as colunas representam diferentes amostras. Cada linha apresenta os escores de um elemento nas *k* amostras. Os escores são então convertidos em postos, variando de 1 a k dentro de cada linha, refletindo o desempenho dos elementos sob cada amostra (SIEGEL; CASTELLAN JR, 2006). Na aplicação, as linhas são os logs de eventos (elementos das amostras) e as colunas os algoritmos (as amostras).

Siegel e Castellan jr (2006) apresentam o passo a passo para aplicação do teste de *Friedman* conforme descrito abaixo:

1. Organizar os dados em uma matriz com N linhas (amostras) e k colunas (variáveis/elementos das amostras).
2. Designar postos aos dados em cada linha, variando de 1 a k.
3. Calcular a soma dos postos para cada coluna (R_j).
4. Utilizar a Equação 17 para computar o valor de *Fr* em casos sem empates, ou recorrer à Equação 18 quando houver empates nas linhas.

Equação 17: Cálculo *Fr* sem empates.

$$Fr = \left[\frac{12}{Nk(k+1)}\sum_{j=1}^{k} R_j^2\right] - 3N(k+1)$$

Equação 18: Cálculo *Fr* com empates.

$$Fr = \frac{12\sum_{j=1}^{k} R_j^2 - 3N^2k(k+1)^2}{Nk(k+1) + \left(Nk - \sum_{i=1}^{N}\sum_{j=1}^{gi} t_{i,j}^3\right)}$$

Onde,

k: número de colunas (amostras);

N: número de linhas (elementos de cada amostra);

R_j: soma dos postos na *j*-ésima coluna (isto é, a soma dos postos para a *j*- ésima variável);

g_i: o número de conjuntos de postos empatados no *i*-ésimo grupo;

t_{ij}: o tamanho do *j*-ésimo conjunto de postos empatados no *i*-ésimo grupo;

Rank médio: $\frac{Rj}{N}$.

5. Estabelecer a probabilidade de um valor observado de *Fr* ser encontrado sob a hipótese *nula*, H0, o que varia conforme os tamanhos de *N* e *k*:
    a. Consultar a tabela de valores críticos (*Fr*) para a análise de variância de dois fatores de Friedman pela estatística de postos para amostras pequenas de *N* e *k*.

b. Para amostras maiores de *N* e/ou *k*, utilizar a distribuição $\chi 2$ (*qui-quadrado*) com graus de liberdade igual a k – 1 para determinar a probabilidade associada.

6. **Rejeitar H0** se a probabilidade determinada no passo 5 for **menor ou igual** ao nível de significância α.

Quando o valor obtido para *Fr* é significante, ele indica que pelo menos uma das condições difere de pelo menos uma outra condição. Ele não informa ao qual delas é diferente, nem quantos grupos são diferentes uns dos outros (SIEGEL; CASTELLAN JR, 2006).

3.3.2 Teste de *Nemenyi*

Caso a hipótese nula do teste de *Friedman* seja refutada, é adequado seguir com um teste post-hoc. O teste de *Nemenyi*, introduzido em 1963, compara todos os algoritmos/amostras entre si. A diferença no desempenho entre dois algoritmos ou amostras é considerada estatisticamente significativa se a distância entre seus *ranks* médios atingir ou superar a diferença crítica (CD) cf. a Equação 19 e com bases nos valores da Tabela 13 (DEMŠAR, 2006).

Equação 19: Diferença crítica do Teste de *Nemenyi*.

$$CD = q_{\propto}\sqrt{\frac{k(k+1)}{6N}}$$

Onde,

k: número de amostras;

N: número de elementos de cada amostra;

$q_{\propto}$: valor crítico para Teste de *Nemenyi* bi-caudal.

Tabela 13: Valores críticas para Teste de *Nemenyi* bi-caudal.

Nº amostras	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$q_{0,05}$	1,960	2,343	2,569	2,728	2,850	2,949	3,031	3,102	3,164
$q_{0,10}$	1,645	2,052	2,291	2,459	2,589	2,693	2,780	2,855	2,920

Fonte: Demšar (2006)

3.4 Considerações do Capítulo

O capítulo explorou a geração e validação de logs de eventos com *drifts*. E recapitulando, os *drifts* na perspectiva do fluxo/estrutura em processos de negócio são, por exemplo, variações na estrutura do processo de negócio que afetam a execução e o desempenho dos processos, podendo comprometer a eficiência operacional e os resultados finais. Existem quatro tipos principais de *drifts* em processos de negócio: abrupto, recorrente, incremental e gradual. Cada um com características distintas, a saber: *drift* abrupto ocorre de maneira repentina e imediata, causando uma mudança abrupta na execução do processo. Geralmente, é desencadeado por eventos inesperados, como falhas em sistemas ou alterações regulatórias, que resultam em interrupções ou desvios significativos nos procedimentos padrão; *drift* recorrente é caracterizado por mudanças que ocorrem periodicamente ao longo da execução do processo. Essas variações podem estar relacionadas a sazonalidades, como picos de demanda em determinadas épocas do ano, ou a padrões cíclicos, como ciclos de produção em indústrias sazonais; *drift* incremental envolve mudanças graduais e contínuas na execução do processo ao longo do tempo. Geralmente, é resultado de ajustes progressivos nas operações ou nos requisitos do mercado. Por exemplo, mudanças nas preferências dos clientes podem levar a ajustes graduais nos processos de produção ou nas estratégias de *marketing*; e *drift* gradual também envolve mudanças suaves e contínuas na execução do processo ao longo do tempo. No entanto, ao contrário do *drift* incremental, o *drift* gradual pode não ter uma direção clara e pode ser mais difícil de detectar.

A investigação para compreender e monitorar esses diferentes tipos de *drifts* em processos de negócio é crucial para garantir a adaptação contínua e a eficácia operacional, permitindo que as organizações identifiquem e respondam prontamente a mudanças no ambiente empresarial.

As diferentes estratégias de geração de logs de eventos artificiais com *drifts*, *ora* concebidas e colocadas em prática neste capítulo, são mostradas em operação, por meio de uma etapa de experimentação, no Capítulo 4.

4 LOGS DE EVENTOS GERADOS PELO PDG

A experimentação visa avaliar o artefato principal do projeto, o sistema de geração de logs de eventos artificiais com diferentes tipos de *drifts* em processo de negócio, denominado **PDG** (*Process Drift Generator*). E, neste contexto, os experimentos foram divididos em dois momentos:

1. Reprodução, utilizando o **PDG**, e validação de logs de eventos sintéticos da literatura repetitivamente; e
2. Geração de novas base de dados com os diversos tipos de *drifts*, utilizando o **PDG**, e validação dos logs de eventos.

Para esses dois momentos, foi delineada a geração de dois conjuntos de logs de eventos, denominados respectivamente de "**Base Gerada 1**" e "**Base Gerada 2**". A primeira base é gerada por meio da reprodução de um conjunto de dados previamente descrito na literatura. Neste processo, busca-se reproduzir o referido conjunto de logs com a maior fidelidade possível, utilizando a mesma parametrização. A similaridade entre a base original e sua reprodução serve como indicador do desempenho do **PDG** na correta geração de logs de eventos com *drifts*. Por sua vez, a geração da segunda base envolve a aplicação direta do **PDG** para gerar *novos* conjuntos de logs de eventos com *drifts* válidos. Essa etapa visa não apenas testar a capacidade do **PDG** de produzir logs de eventos com variações, mas também validar os logs.

Em suma, os conjuntos de logs usados nos experimentos estão descritos na Tabela 14.

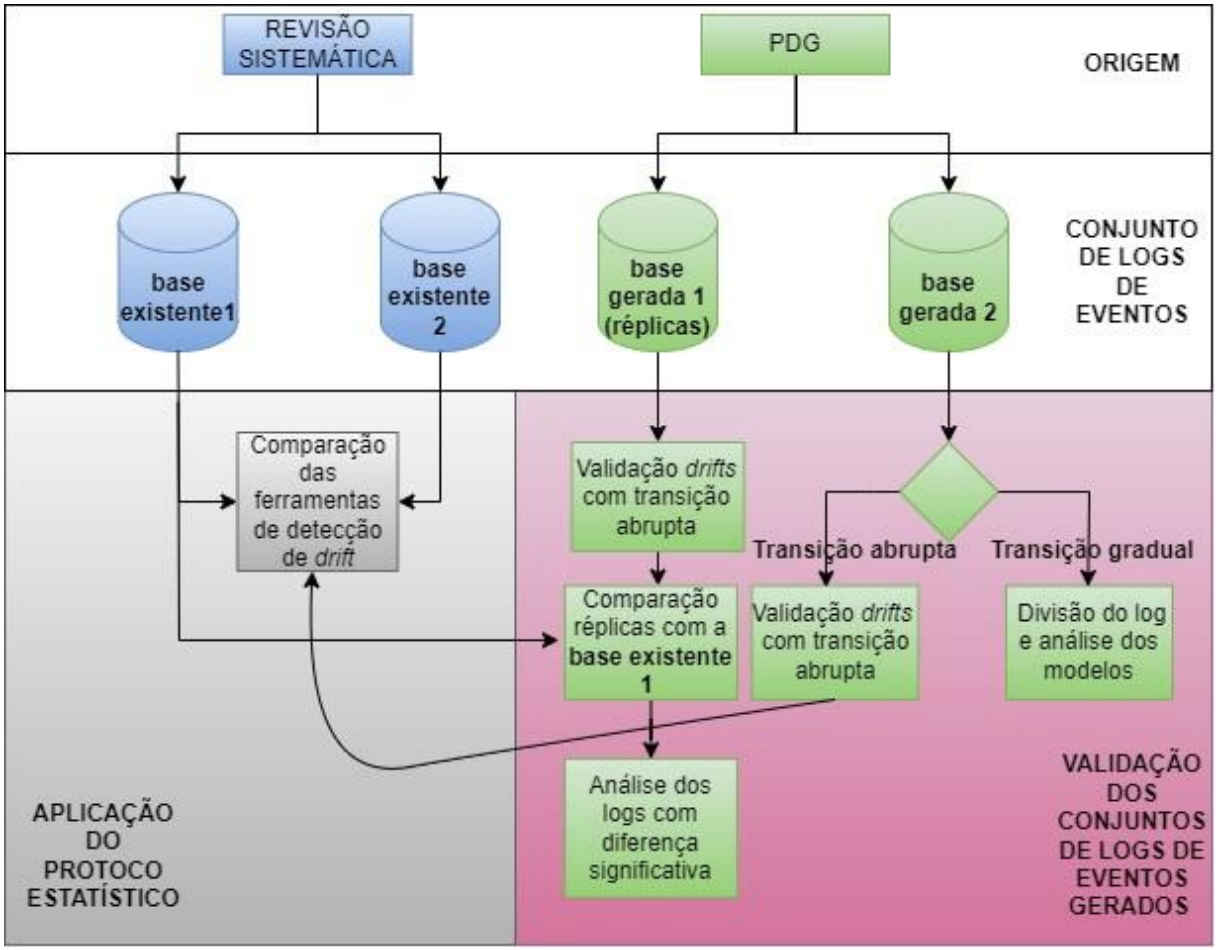
Tabela 14: Bases existentes (1 e 2) e bases geradas (1 e 2).

Conjunto de logs de eventos	Descrição	Objetivo	Seção
Base Existente 1	Conjunto de 18 logs extraídos da **base de logs de eventos artificiais existente 3 (BLA3**)	Comparar ferramentas existentes detecção de *drifts*.	4.1
Base Existente 2	Conjunto de 15 logs extraídos da **base de logs de eventos artificiais existente 4 (BLA4)**	Comparar ferramentas existentes de detecção de *drifts*.	4.1
Base Gerada 1 (reproduzida)	Réplica da **Base Existente 1** gerada pelo método **PDG** com 17 dos 18 logs existentes.	Validar conformidade dos logs gerados pelo método **PDG** para a criação de logs com *drifts*.	4.2
Base Gerada 2	Conjunto de logs de eventos com diferentes tipos de *drifts* gerado pelo método **PDG**	Avaliar o gerador de logs de eventos com *drifts* em operação e comparar ferramentas de detecção de *drift*	4.3

Fonte: O autor (2024)

O **PDG** foi empregado na criação de duas bases de dados: a **Base Gerada 1** (*reproduzida*) e a **Base Gerada 2**. A **Base Gerada 1** inclui exclusivamente *drifts* com transições abruptas, sendo submetida inicialmente ao processo de validação proposto na Subseção 3.2.2 e, em seguida, comparada à **Base Existente 1**. Por outro lado, a **Base Gerada 2** é constituída a partir de parâmetros ainda não explorados na literatura. Os *drifts* com transição abrupta nesta base são validados conforme descrito na Subseção 3.2.2 enquanto os *drifts* com transição gradual são tratados conforme especificado na Subseção 3.2.3. Posteriormente, as bases já existentes (**Base Existente 1** e **Base Existente 2**) e a **Base Gerada 2** são empregadas para comparação com as ferramentas de detecção de *drift*, utilizando o protocolo de análise estatística descrito na Seção 3.3. O fluxo que descreve a origem, o processo de validação e a correlação entre os conjuntos de logs de eventos é ilustrado na Figura 31.

Figura 31: Apresentação dos conjuntos de logs de eventos utilizadas para validação do PDG e comparação das ferramentas de detecção de *drift*.



Fonte: O autor (2024)

As bases de logs de eventos foram definidas da seguinte forma:

- **Base Existente 1**: Adicionalmente, ao que já foi dito na Tabela 14, a **Base Existente 1** contém 18 logs de eventos extraídos da **base de logs de eventos artificias existentes/literatura 3 (BLA3),** Maaradji et al. (2015), cada log de eventos possui 5.000 traços e 9 *drifts* abruptos. Foram escolhidos os 18 logs com 5000 traços, pois foram os menores logs em que foram reportados em todos os testes em Maaradji et al. (2015).
    - Resumo: 5.000 traços, 18 logs de eventos, 9 *drifts* abruptos.
- **Base Existente 2:** A base possui 15 logs de eventos extraídos da **base de logs de eventos artificias existentes/literatura 4** (**BLA4**), Ceravolo et al. (2020), cada log de eventos possui 1.000 traços, 1 *drift* abrupto e 0% de ruído.
    - Resumo: 1.000 traços, 15 logs de eventos, 1 *drift* abrupto.
- **Base Gerada 1 (**reproduzida**):** Deste modo, 17 logs da **Base Existente 1** foram reproduzidos 10 vezes e deram origem a **Base Gerada 1**. Essas amostras serão usadas para mostrar a corretude da ferramenta **PDG.**
    - Resumo: 5.000 traços, 170 logs de eventos, 9 *drifts* abruptos.
- **Base Gerada 2:** Empregando o **PDG** foram gerados 20 logs de eventos com diferentes números de traços, *drifts* recorrentes, incrementais, abruptos e graduais, presença de ruído.
    - Resumo: 21 logs de eventos com diferentes parâmetros incluindo ruído, intervalos variáveis entre os *drifts*, diferentes tamanhos de logs de eventos, os quatro tipo de *drifts* e logs com mais de um tipo de *drift*.

4.1 Geração e validação da Base Gerada 1

Os parâmetros de geração da **Base Gerada 1 (**reproduzida**)** serão apresentados e, em sequência, será feita a validação dos logs de eventos.

4.1.1 Geração da Base Gerada 1

A **Base Gerada 1 (**reproduzida**)** é composta por 17 conjuntos de logs, onde cada conjunto de logs instância um padrão de mudança. Cada um dos 17 padrões da **Base Existente**

1 foi reproduzido 10 vezes. A primeira linha/conjunto da **Base Gerada 1 (**reproduzida**)**[39] representa 10 reproduções do padrão cb5k. Deve-se notar que cada elemento da **Base Existente 1** (abaixo) representa um log de eventos com 5.000 traços e o nome do padrão de mudança — dado pelas duas primeiras letras no referido nome — constante no log de eventos.

Base Existente 1 =[cb5k, cd5k,..., rp5k, sw5k]

Base Gerada 1 (reproduzida**)** ={cb5k: [cb5k$_1$, cb5k$_2$, ..., cb5k$_{10}$],
cd5k: [cd5k$_1$, cd5k$_2$, ..., cd5k$_{10}$],
...
rp5k: [rp5k$_1$, rp5k$_2$, ..., rp5k$_{10}$],
sw5k: [sw5k$_1$, sw5k$_2$, ..., sw5k$_{10}$]}

Deve-se notar que foram utilizados 17 dos 18 padrões de mudança de Maaradji et al. (2015); apenas o padrão ***fr*** foi deixado de fora dado que ele não é uma mudança estrutural. Trate-se apenas de uma mudança na frequência das atividades. Deste modo, o *fr5k* foi retirado da **Base Existente 1** nesse momento. Adicionalmente, os logs lp2.5k e re2.5k continham 5000 traços e foram utilizados na **Base Existente 1** e **Base Gerada 1 (**reproduzida**)**. Para facilitar, eles foram renomeados para lp5k e re5k (o mesmo procedimento foi feito em Sato (2022)). Assim, o conteúdo dos dois conjuntos de logs de eventos são:

- **Base Existente 1:** 17 padrões de mudança, 17 logs de eventos (o padrão *fr* foi excluído nesse momento).
- **Base Gerada 1 (**reproduzida**):** 17 padrões de mudança, 170 logs de eventos.

4.1.2 Validação da Base Gerada 1

O processo de validação foi feito conforme a Seção 3.2.4, as configurações dos logs de eventos foram replicadas 10 vezes para cada uma na **Base Gerada 1 (**reproduzida**)** e seguiu o fluxo de proposto na Figura 28. Sobre a comparação dos resultados obtidos, executou-se o teste de *Wilcoxon* para duas amostras pareadas com N igual a 8, sendo 8 as janelas em que cada base foi testada (janela adaptativa com janela inicial de 32, 50, 75, 100, 125, 150, 162, 200 traços). Portanto, o Algoritmo 7, apresentado em 3.2.4, é utilizado para comparação dos *drifts*

[39] https://www.kaggle.com/conjuntos de logs de eventos/caioraduy/replicating-existent-event-logs-using-loggen

detectados *log-a-log*, considerando ambas as bases de logs de eventos: **Base Existente 1** e **Base Gerada 1 (**reproduzida**)**.

Em suma, aplicaram-se tais comparações de desempenho sobre os 17 logs de eventos da **Base Existente 1** com a **Base Gerada 1 (**reproduzida**)**, resultando em 170 testes estatísticos para cada algoritmo de detecção de *drifts*. Os algoritmos de detecção de *drifts* usados foram: IPDD *Framework* e Apromore-*ProDrift*, e ambos parametrizados para operar na abordagem adaptativa; gerou-se no final do processo 340 testes.

Desta forma, tomando como exemplo o log sw5k, o teste de *Wilcoxon* usa os valores obtidos para o *F-score* presentes na Tabela 15, essa tabela mostra apenas os resultados obtidos para os 17 logs da **Base Existente 1**. A Tabela 16 mostra apenas os 17 logs gerados na quinta replicação da **Base Gerada 1 (**reproduzida**);** conteúdo deve-se notar que a **Base Existente 1** foi replicada 10 vezes. Neste contexto, o teste de *Wilcoxon* foi aplicado entre o conjunto do *F-score* original sw5k = {1, 1, 1, 1, 1, 1, 1, 1} e a quinta réplica de sw5k = {1, 1, 1, 1, 1, 1, 1, 1}. Vale citar que a amostra com todos os elementos/resultados iguais, não permite aplicar o teste em questão, e neste caso, devem ser considerados iguais. Logo, a 5ª réplica representou um "acerto", ou seja, tal réplica foi significativamente igual ao log original e ela se encontra registrada na Tabela 17. Deve-se notar que para cada log, o procedimento de réplica gerou 10 réplicas.

A Tabela 17 resume os resultados na análise estatística e mostra os números de comparações em que os resultados foram significativamente iguais. No caso da reprodução do log de eventos sw5k, para o Apromore-*ProDrift-Adaptive* 9 das 10 (ou 9/10) amostras foram significativamente iguais, obteve-se um *p-valor* maior que 0,05 para o teste de *Wilcoxon*, e para o *Adaptive* IPDD *trace by trace* 10 das 10 amostras (ou 10/10). Os resultados do *F-score*s para a **Base Gerada 1 (**reproduzida**)** estão disponíveis no *kaggle*[40].

[40] https://www.kaggle.com/datasets/caioraduy/f-score-base-gerada1

Tabela 15: *F-score* para os 8 testes realizados no *Adaptive* IPDD *trace by trace* para a Base Existente 1.

Janela inicial	32	50	75	100	125	150	162	200
cb5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
cd5k	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
cf5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
cm5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
cp5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
IOR5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
IRO5k	0,89	0,89	0,89	0,89	0,89	0,89	0,89	0,89
lp5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
OIR5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
ORI5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
pl5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
pm5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
re5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
RIO5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
ROI5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
rp5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
sw5k	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Fonte: O autor (2024)

Tabela 16: *F-score* para os 8 testes realizados no *Adaptive* IPDD *trace by trace* para a quinta réplica da Base Gerada 1.

Janela inicial	32	50	75	100	125	150	162	200
cb5k_PDG_5	0,56	0,56	0,56	0,56	0,56	0,56	0,56	0,56
cd5k_PDG_5	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
cf5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
cm5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
cp5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
IOR5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
IRO5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
lp5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
OIR5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
ORI5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
pl5k_PDG_5	0,78	0,78	0,78	0,78	0,78	0,78	0,78	0,78
pm5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
re5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
RIO5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
ROI5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
rp5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
sw5k_PDG_5	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Fonte: O autor (2024)

Tabela 17: Número de acertos por log para o Apromore e o IPDD.

	Abordagem Adaptativa	
Log	Apromore-*ProDrift Adaptive*	*Adaptive* IPDD *trace by trace*
cb5k	5/10	2/10
Cd5k	10/10	10/10
cf5k	10/10	10/10
cm5k	8/10	10/10
cp5k	10/10	10/10
IOR5k	10/10	10/10
IRO5k	10/10	10/10
lp5k	8/10	10/10
OIR5k	10/10	10/10
ORI5k	10/10	10/10
pl5k	10/10	10/10
pm5k	8/10	10/10
re5k	10/10	10/10
RIO5k	10/10	10/10
ROI5k	9/10	10/10
rp5k	10/10	10/10
sw5k	9/10	10/10

Fonte: O autor (2024)

Inicialmente, o experimento de validação foi feito apenas no Apromore-*ProDrift*. E na expectativa de ampliar a confiança da validação, utilizou-se também o IPDD *framework.* Portando, levando em conta os resultados obtidos com os dois detectores, apenas o log cb5k não obteve 10/10 significativamente iguais no primeiro log (cb5k) conforme será analisado em sequência (cf. Figura 32, Figura 33 e Tabela 18).

Os resultados para 16 dos 17 logs de eventos atenderam às expectativas, em que no mínimo 8 das 10 reproduções foram estatisticamente iguais. Além disso, no IPDD, 16 dos 17 logs obtiveram 100% de assertividade.

Para confirmar a presença do *drift* nos logs em que foram obtiveram resultados inferiores a 10/10, foi feito o seguinte passo a passo:

- Dividir o log em sub-logs de acordo com a posição no *drift*, nesse caso, a cada 500 traços;
- Submeter o log a uma ferramenta de descoberta para visualização dos modelos;
- Comparar visualmente os modelos;
- Analisar o *F-score* das amostras.

Deste modo, em todos os padrões que não obtiveram resultados iguais a 10/10 em ambas ferramentas, foi feita a investigação dos modelos que os modelos estavam de fato sendo gerado com a estrutura/fluxo do modelo correta.

A investigação mostrou que os modelos são gerados corretamente, no entanto, com diferenças nas frequências das variantes/caminhos. Isso acontece, porque, o método original, conforme Maaradji et al. (2015), utiliza o BIMP *Simulator* em que é possível especificar a probabilidade dos caminhos, enquanto, que a função escolhida do BASIC.PLAYOUT do PM4PY com a entrada do modelo de processo em PNML não é possível. No **APÊNDICE A- Extração dos modelos da Base Gerada 2 (reproduzida)**, os modelos de um log de cada padrão são reportados e evidenciam a presença do *drift*. A seguir, como exemplo, o processo de validação será feito para o padrão cb.

Tomando como exemplo a diferença de frequência das atividades do log original (cb5k) para o log réplica/reproduzido de cb5k gerado pelo **PDG**, a Figura 32 mostra que no log de eventos cb5k (lado esquerdo), 58 traços passam pela atividade "*Return application back to applicant*" e na reprodução do mesmo cb5k (lado direito) passam 558 traços. Adicionalmente, a atividade "*Reject application*" é registrada 271 vezes em cb5k (lado esquerdo) e 183 vezes na reprodução de cb5k (lado direito), conforme mostra a Figura 33. As diferenças em porcentagem das frequências das variantes são ilustradas/indicadas nas Figura 32 e Figura 33.

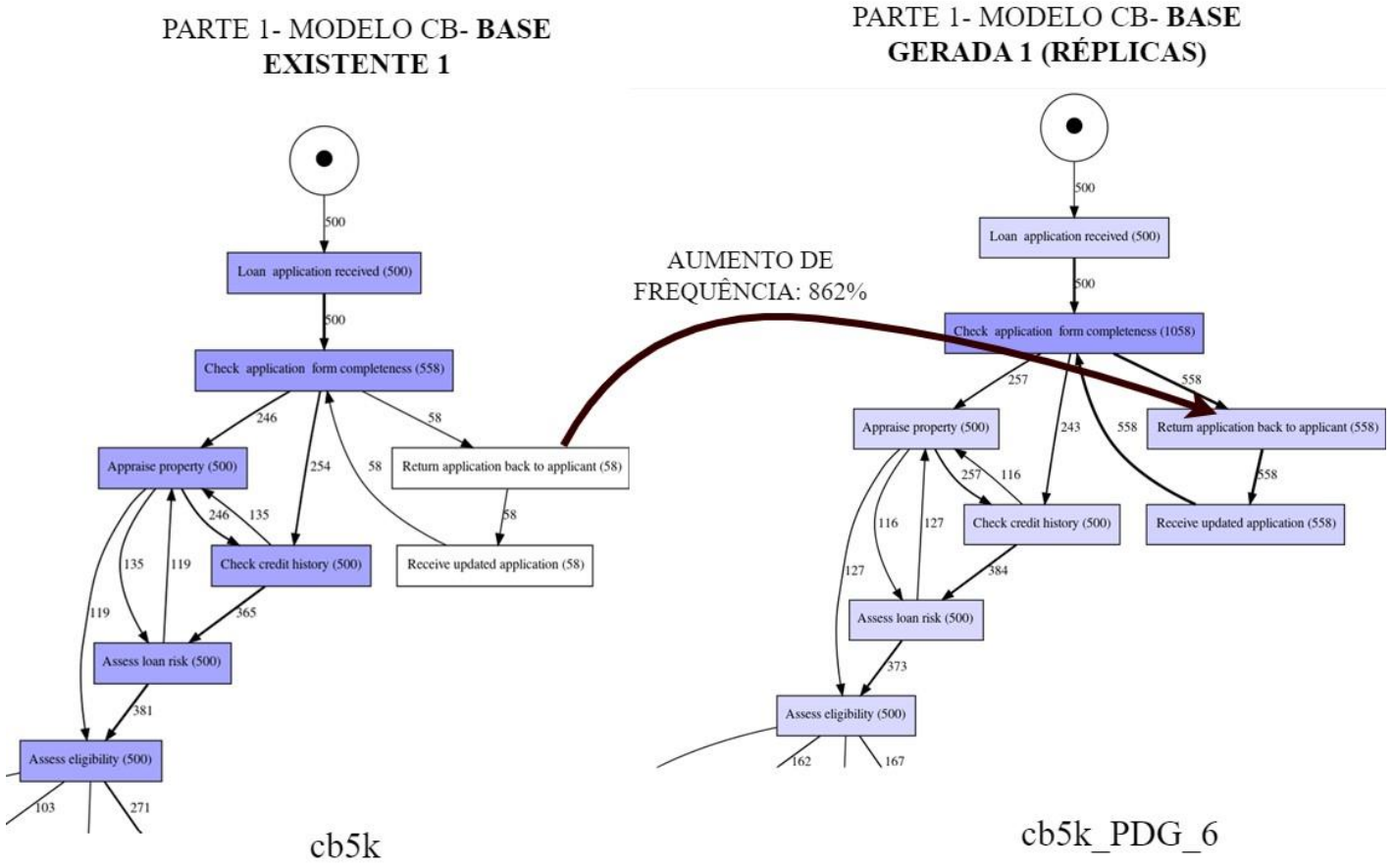
Em seguida, foi feita a análise do *f-score* com os *drift* relatados/detectados e os *drifts* reais, onde observou-se que em vários casos os *drifts* são detectados. No entanto, pelo fato da utilização do *F-score* com tolerância (reativo), os *drifts* detectados após o erro de tolerância não entram nas contas do número TP. Deste modo, eles são considerados FPs e alteram de maneira significativa o *F-score*. Tomando como exemplo a segunda linha da Tabela 18, o cb5k_PDG_6 submetido ao *Adaptive* IPDD *trace by trace*, encontra 8 dos 9 *drifts*, no entanto, 4 dos *drifts* encontrados estão fora da janela de tolerância. Os *drifts* detectados são [511, 1343, 1535, 2111, 2527, 3135, 3519, 4447] e isso gera um *F-score* igual a 0,36. Além disso, na segunda linha da Tabela 18, todos os seguintes *drift*s são detectados no log cb5k: [511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]. Porém, com o erro de tolerância igual ao tamanho da janela, o *F-score* obtido é 0,33. Essas oscilações no *F-score* são responsáveis pelas diferenças estatísticas encontradas.

Em suma, as diferenças obtidas entre os logs originais e os logs reproduzidos pelo **PDG** acontecem devido aos geradores de traços básicos: a base original Maaradji et al. (2015) utiliza o BIMP *Simulator* e a **Base Gerada 1 (**reproduzida) utiliza o gerador do PM4PY. Os simuladores geram traços com frequências diferentes, como foi constatado, e isso influencia diretamente na posição do *drift* detectado pelo gerador. Uma vez que o *F-score* é calculado com

uma janela de tolerância igual ao tamanho da janela inicial, qualquer diferença na posição do *drift* gera um grande impacto no *F-score* conforme é possível observar na Tabela 18.

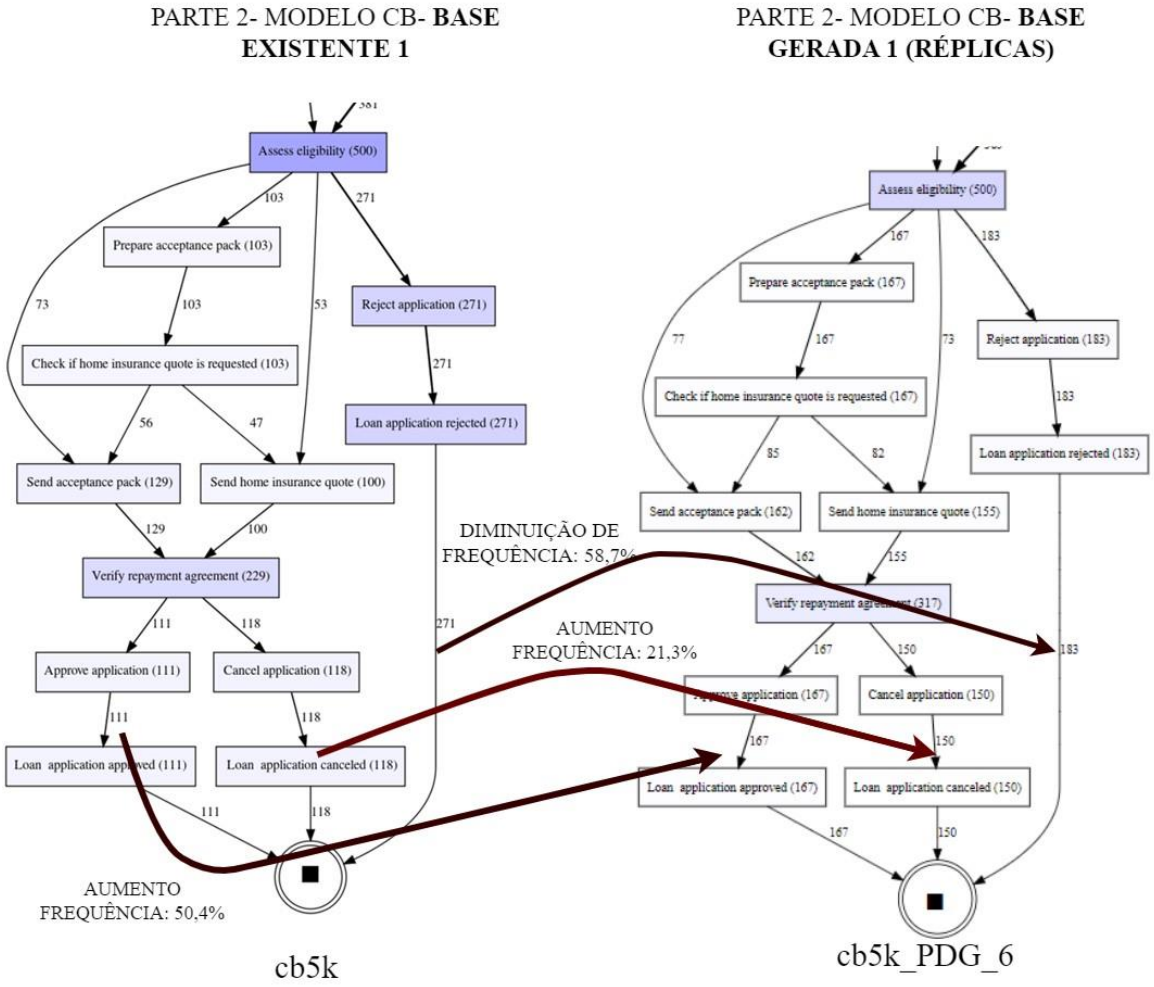
Na próxima Seção serão apresentados os detalhes da **Base Gerada 2**.

Figura 32: Diferença de frequências (parte 1 do modelo cb nos logs de eventos cb5k e cb5k_PDG_6).



Fonte: O autor (2024)

Figura 33: Diferença de frequências (parte 2 do modelo cb nos logs de eventos cb5k e cb5k_PDG_6).



Fonte: O autor (2024)

Tabela 18: Comparação *F-score* (cb5k x cb_PDG_6).

Ferramenta	Log de eventos	Janela	Janela inicial	*F-score*	Posição *drifts d*etectados
IPDD	cb5k	*adaptive*	32	0,333333	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	32	0,352941	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	50	0,555556	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	50	0,470588	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	75	0,888889	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	75	0,470588	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	100	1	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	100	0,470588	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	125	1	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	125	0,588235	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	150	1	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	150	0,705882	[511, 1343, 1535, 2111, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	162	1	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	162	0,5	[511, 1343, 2175, 2527, 3135, 3519, 4447]
IPDD	cb5k	*adaptive*	200	1	[511, 1055, 1535, 2079, 2527, 3071, 3519, 4063, 4543]
IPDD	cb5k_PDG_6	*adaptive*	200	0,625	[511, 1343, 2175, 2527, 3135, 3519, 4447]

Fonte: O autor (2024)

4.2 Geração e validação da Base Gerada 2 – múltiplos logs e tipos de *drifts*

Os parâmetros de geração da **Base Gerada 2** serão apresentados e, em sequência, será feita a validação dos logs de eventos.

4.2.1 Geração da Base Gerada 2

Os experimentos foram realizados com base nos modelos de processos disponibilizados em Maaradji et al. (2015). Devido ao viés dos algoritmos de mineração de processos, os modelos de processos foram reproduzidos manualmente para assegurar a integridade do modelo de processo. Por exemplo, ao extrair o processo de modelo de um log de eventos, o *inductive miner*[41] não consegue reportar atividades repetidas.

Por uma questão de simplificação da apresentação dos resultados/testes, optou-se, *de um lado*, em gerar a **Base Gerada 2** com apenas parte dos modelos, a saber: *modelo base*, *cb*, *lp*, *OIR*, *IOR*, *re*, *sw*, e *de outro lado*, em mostrar o potencial da ferramenta **PDG** em combinar modelos para gerar diferentes variações dos logs de eventos com *drifts*.

Adicionalmente, deve-se notar que foram criados 3 modelos específicos (Modelo I, modelo IO, modelo IORI) para a criação do log com *drifts* incrementais. Os três modelos são utilizados sequencialmente para gerar *drifts*, e eles devem ser ordenados para que as mudanças representem pequenas alterações em relação ao modelo anterior:

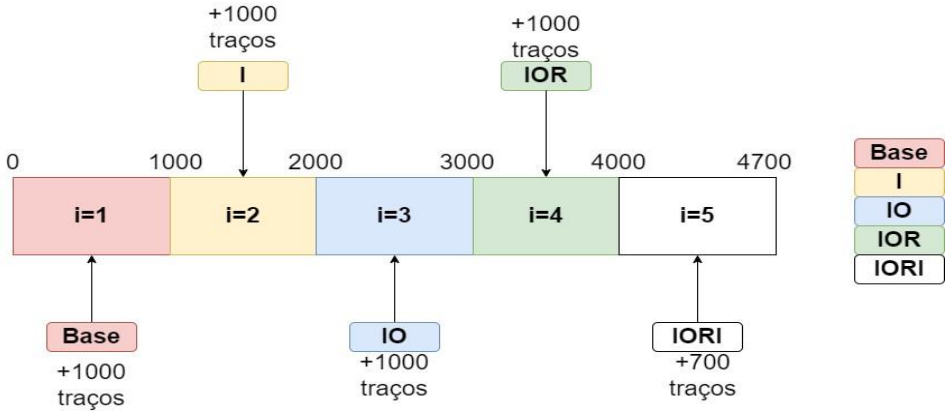
1. Modelo Base (cf. Figura 35)
2. Modelo I: Modelo Base + inserção de atividade (cf. Figura 36)
3. Modelo IO: Modelo I + otimização (cf. Figura 37)
4. Modelo IOR: Modelo IO + re-sequenciamento de atividade (cf. Figura 38)
5. Modelo IORI: Modelo IOR + inserção de atividade (cf. Figura 39).

Neste contexto, para gerar um log de eventos com quatro *drifts* incrementais, requer-se cinco modelos: um log de eventos com 4.700 traços e quatro *drifts* incrementais localizados nos traços 1.000, 2.000, 3.000 e 4.000 (cf. Figura 34). Gera-se então: 1.000 traços do Modelo Base, 1.000 traços do Modelo IO, 1.000 traços do modelo IO, 1.000 traços do modelo IOR e 700

[41] O *inductive miner* é um algoritmo consolidado na descoberta de modelos (https://www.futurelearn.com/info/courses/process-mining/0/steps/15642)

traços do Modelo IORI.

Figura 34: Exemplo de log de eventos sintéticos com *drift*s incrementais.



Fonte: O autor (2024)

A Tabela 19 descreve os padrões de mudanças simples que foram escolhidos para o experimento. Os outros modelos estão disponíveis em conjunto com os logs de eventos. Deve-se notar que tais logs de eventos foram criados pelo **PDG** e testados nos detectores de *drifts* Apromore, IPDD e VDD *system*. A Tabela 20 representa o conjunto de logs sintéticos gerados e seus parâmetros de geração. Os logs sintéticos gerados estão disponíveis *online*[42].

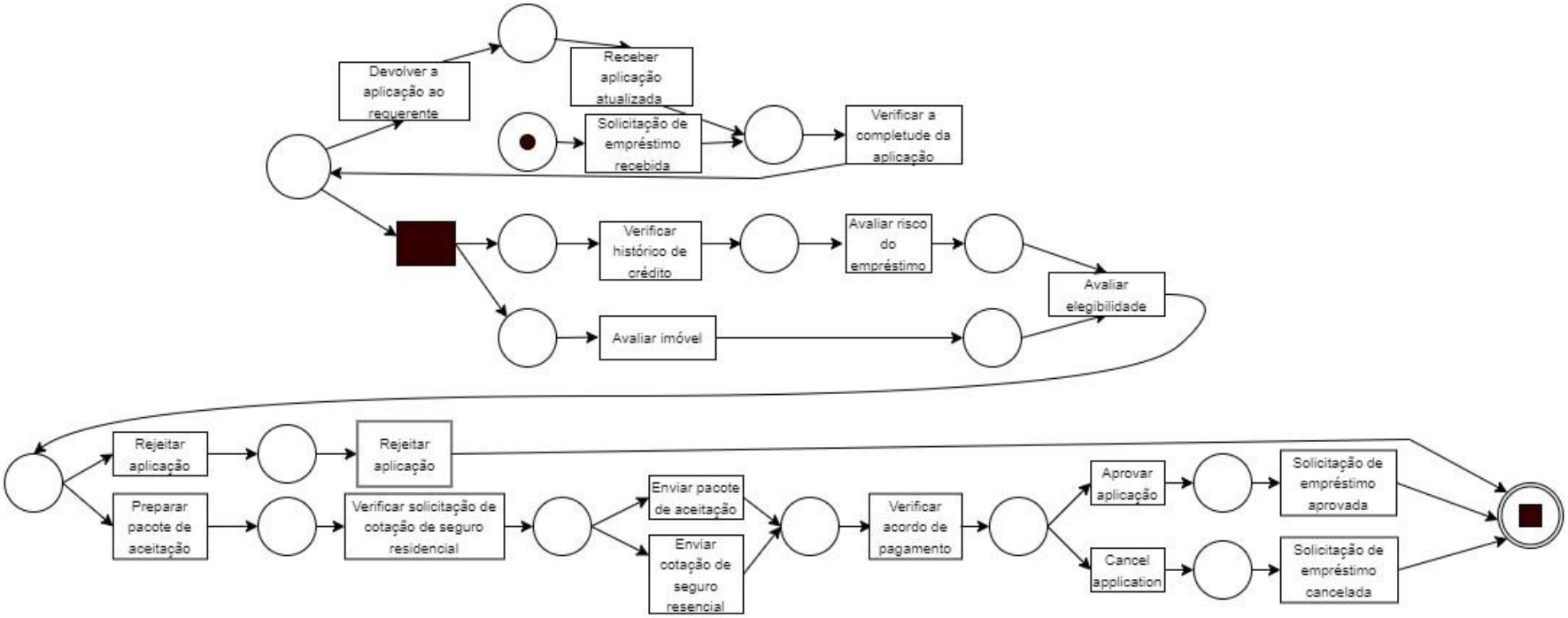
Tabela 19: Parâmetros de geração de logs de eventos mudanças simples.

Código	Padrões de mudanças simples	Categoria
re	Adicionar/remover fragmento	I
lp	Tornar fragmento repetível/não repetível	O
cb	Tornar fragmento opcional/não opcional	O
sw	Trocar dois fragmentos	I

Fonte: Maaradji et al. (2015)

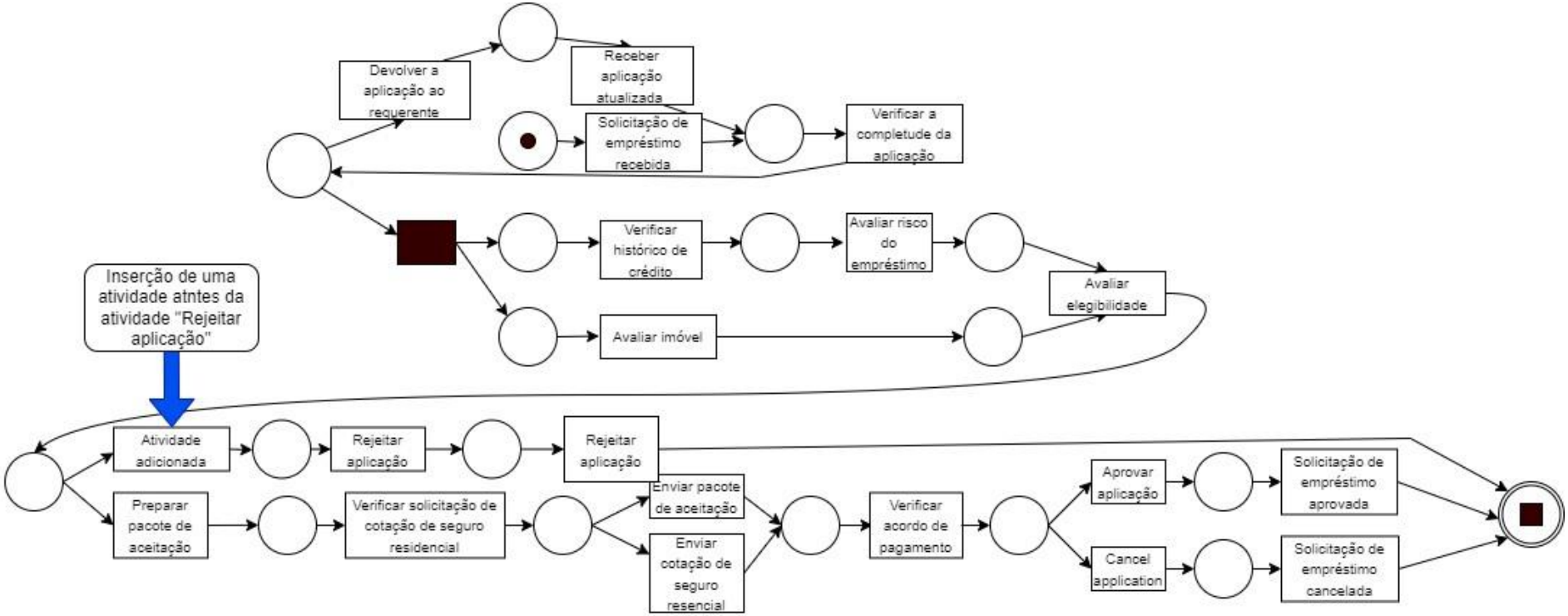
[42] https://www.kaggle.com/datasets/caioraduy/logs-sinteticos-pdg

Figura 35: Modelo base.



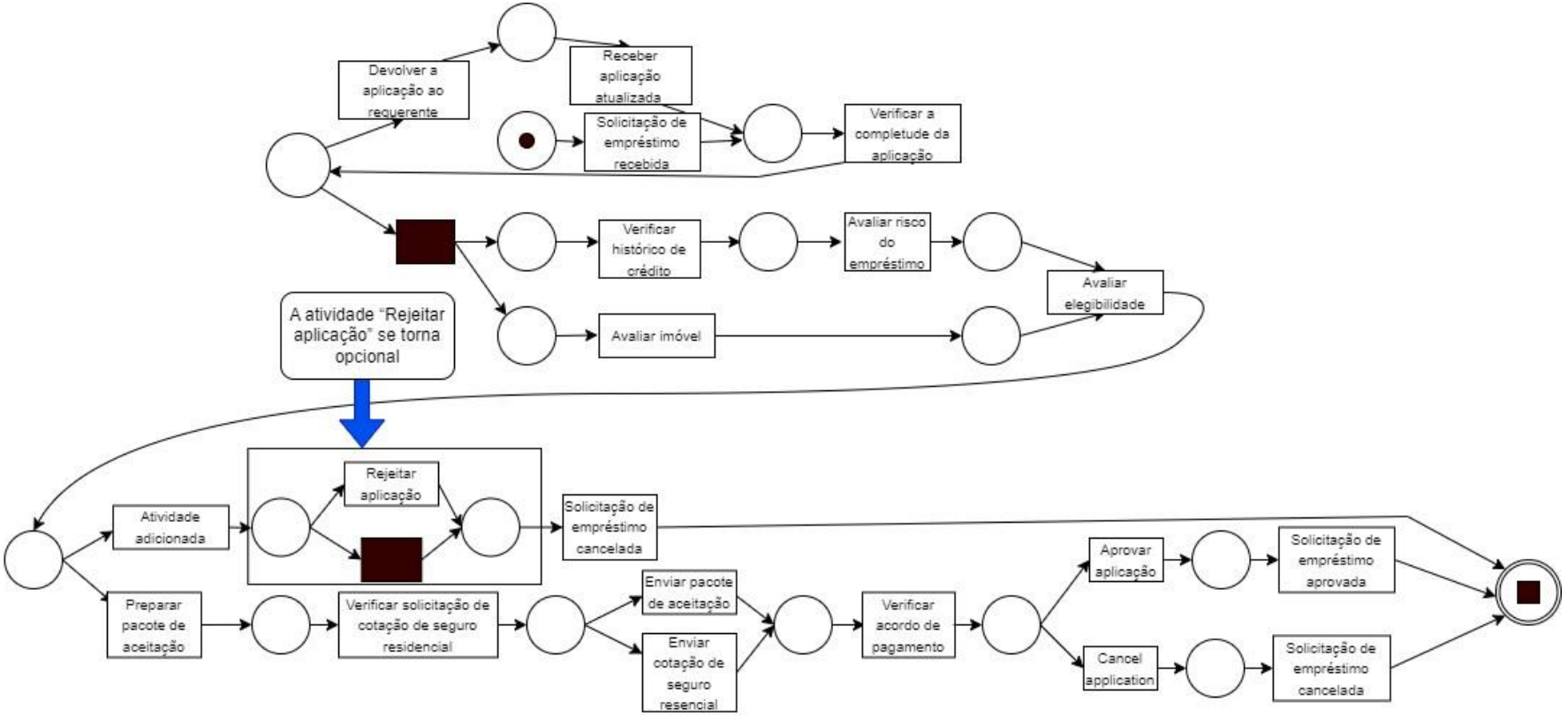
Fonte: O autor (2024)

Figura 36: Modelo I (Modelo base com uma atividade adicionada).



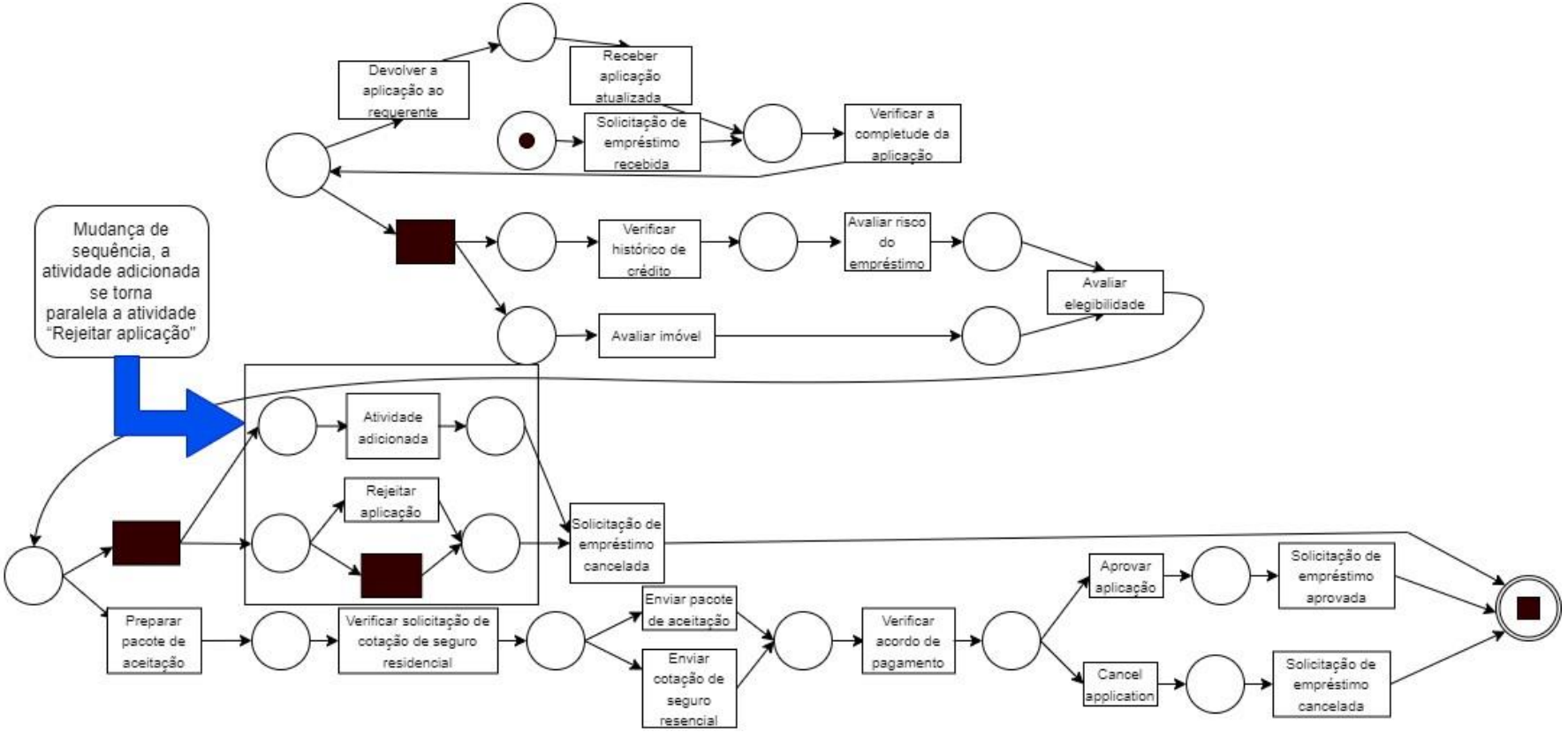
Fonte: O autor (2024)

Figura 37: Modelo IO (Modelo I com otimização).



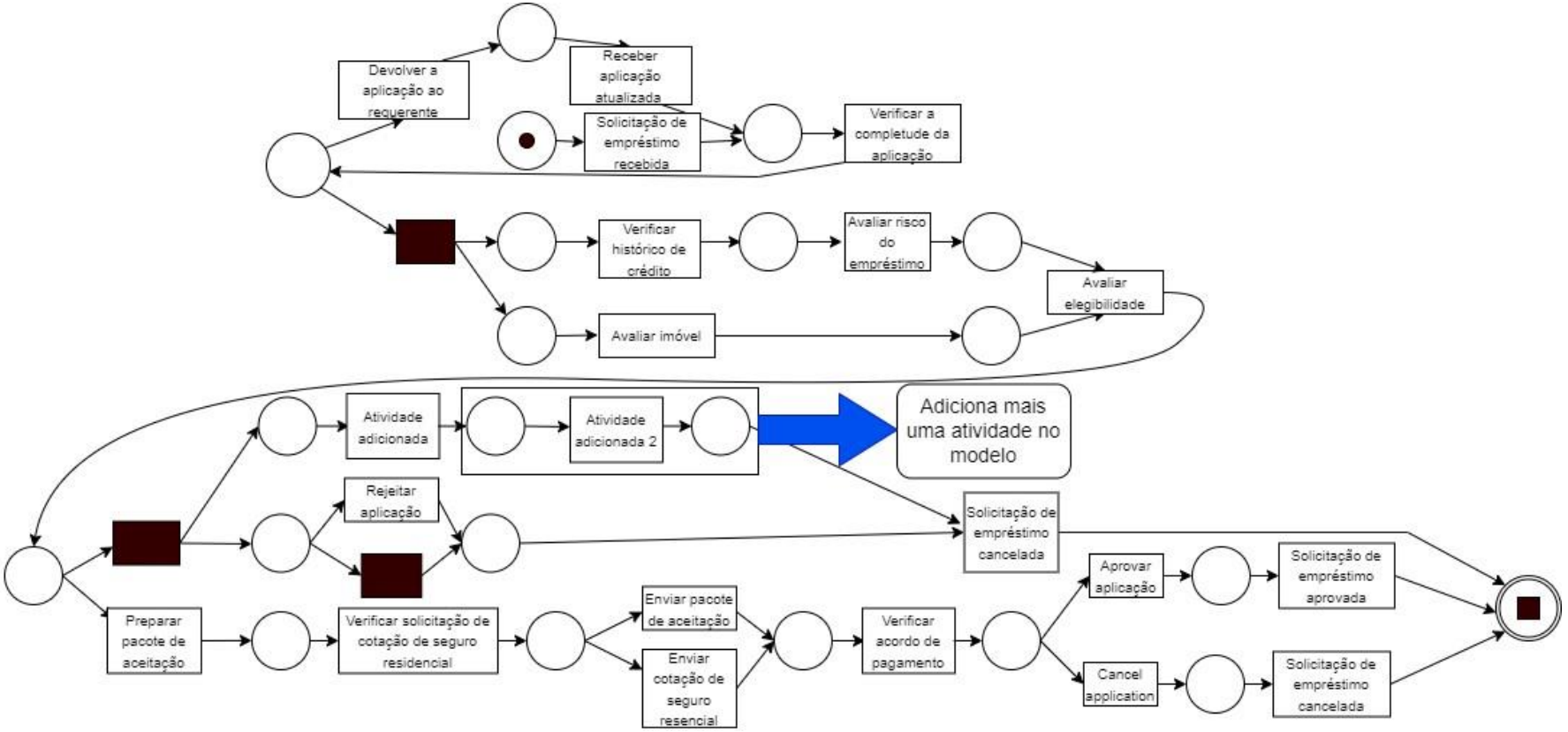
Fonte: O autor (2024)

Figura 38: Modelo IOR.



Fonte: Maaradji et al. (2015)

Figura 39: Modelo IORI (Modelo IOR com atividade adicionada).



Fonte: O autor (2024)

Tabela 20: Logs de eventos sintéticos e seus parâmetros de geração para o Base Gerada 2.

Nome-log	Modelos	Tamanho (traços)	Nº de *drifts*	Localização do *drift*	Decaimento	Ruído (%)	*Timestamp* inicial
LS1-Abrupto	Base, cb, sw	1000	2	300,500	---	---	2021-07-13T10:00:00
LS2-Gradual	Base, cb	1000	2	[300,500]	Linear	---	2021-07-13T10:00:00
LS3-Incremental	Base, I, IO	1000	2	300,500	---	---	2021-07-13T10:00:00
LS4-Recorrente	Base, cb	1000	2	300,500	---	---	2021-07-13T10:00:00
LS5-Gradual	Base, sw	2000	2	[300,500], [1000,1300]	Linear, exponencial	---	2021-07-13T10:00:00
LS6-Incremental	Base, I, IO	2000	2	600,1000	---	---	2021-07-13T10:00:00
LS7-Recorrente	Base, sw	2000	2	600,1000	---	---	2021-07-13T10:00:00
LS8-Abrupto	Base, OIR, sw	3000	3	900,1500,2100	---	---	2021-07-13T10:00:00
LS9-Gradual	Base, OIR	3000	3	[300,500], [1000,1300], [1900,2800]	Linear, exponencial, linear	---	2021-07-13T10:00:00
LS10-Incremental	Base, I, IO, IOR	3000	3	900,1500,2100	---	---	2021-07-13T10:00:00
LS11-Recorrente	Base, OIR	3000	3	900,1500,2100	---	---	2021-07-13T10:00:00
LS12-Abrupto	Base, re, OIR	4300	4	999,1320,2050,3050	---	---	2021-07-13T10:00:00
LS13-Gradual	Base, re	4500	4	[300,500], [800,1200], [2000,2460], [3000,3600]	Linear, exponencial, linear, exponencial	2	2021-07-13T10:00:00
LS14-Incremental	Base, I, IO, IOR, IORI	4700	4	1000,2000, 3000,4000	---	3	2021-07-13T10:00:00
LS15-Recorrente	Base, re	4900	10	2000,3000,4000,5000,6000,7000,8000, 9000,10000,11000,12000,13000, 14000	--_	4	2021-07-13T10:00:00
LS16- Incremental	Base, I, IO, IOR, IORI	15000	14	1000, 2500, 3000, 5000	---	---	2021-07-13T10:00:00
LS17- Gradual	Base, re	5500	3	[500, 800], [1220, 1900], [2000,2460]	Linear, exponencial, linear	4	2021-07-13T10:00:00

LS18- Gradual	Base, I	900	1	[300,600]	Linear	---	2021-07-13T10:00:00
LS19- MergedLog1 (LS15+LS8+LS16)	Base, re, OIR, sw, I, IO, IOR, IORI.	25000	22	2000,3000,4000,5000,6000,7000,8000, 9000,10000,11000,12000,13000, 14000,15000,15900, 16500, 17100, 18000, 1900, 20500, 21000, 23000	---	---	2021-07-13T10:00:00
LS20- MergeLog2 (LS1+LS3+LS4)	Base, cb, sw, I, IO	3000	8	300, 500,1000,1300,1500,2000, 2300, 2500	---	---	2021-07-13T10:00:00
LS21- MergedLog3 (LS7+LS8)	Base, sw, OIR	50000	6	600,1000,2000, 2900, 3500,4100	---	---	2021-07-13T10:00:00

Fonte: O autor (2024)

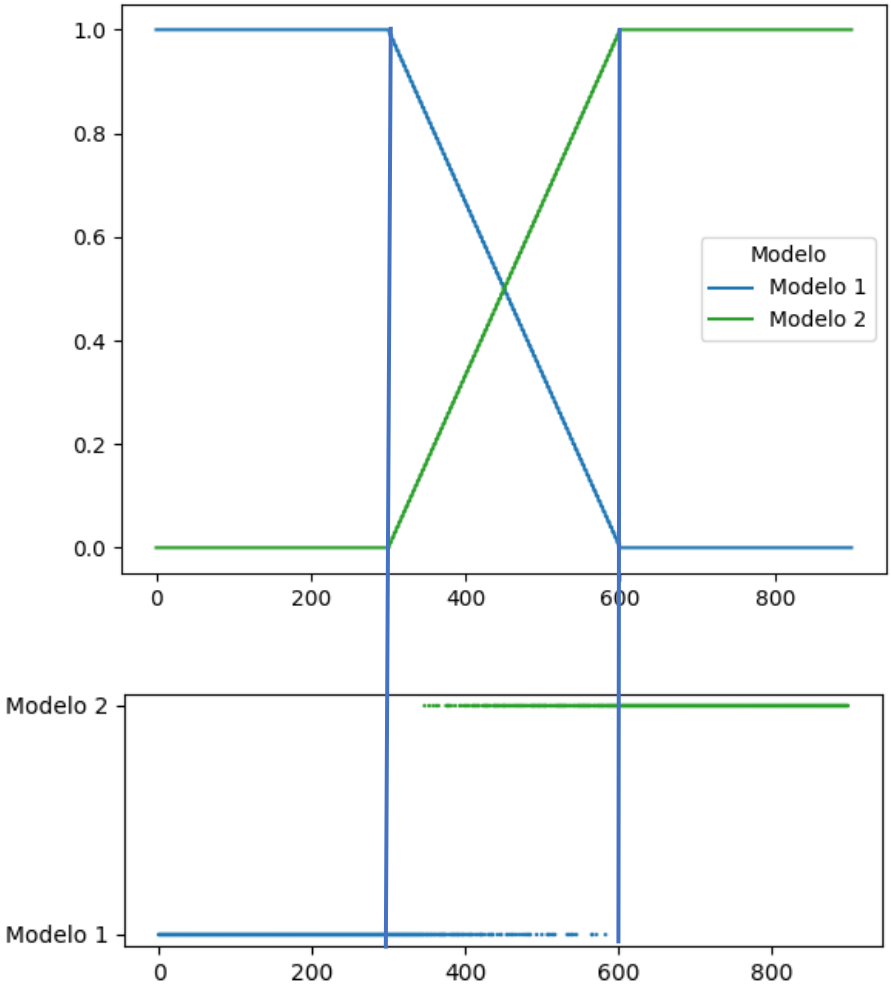
4.2.2 Validação da Base Gerada 2

Como já dito anteriormente, a geração da **Base Gerada 2** = {LS_1, LS_2, LS_3, ..., LS_{21}} evolveu diferentes parâmetros para a geração de logs de eventos com *drifts* LS_i, onde $1 \leq i \leq 21$. Cada comportamento executado pode ser acompanhado pela probabilidade da existência de cada modelo ao longo do log de eventos (cf. Figura 40). E para ilustrar o funcionamento do gerador de logs de eventos, utiliza-se o log de eventos LS_{18}, cuja parametrização é a seguinte.

- Tipo de *drift*: Gradual
- Comportamento: Probabilístico
- Número de traços: 900
- Intervalo de *drift*: [300, 600]
- Log de eventos: LS_{18} (formato XES)

A parte superior da Figura 40 mostra a evolução das probabilidades de cada modelo ao longo da composição do LS_{18}. A parte inferior da Figura 40 ilustra a adição de traços nos modelos 1 e 2 ao longo do log LS_3. Observa-se que, até o traço 300, a probabilidade de encontrar um traço do Modelo 1 é de 100%, e a partir do traço 301, essa probabilidade começa a decair linearmente até atingir zero no traço 600. Em outras palavras, até o traço 300 tem-se traços do Modelo 1 em LS_3, e do 301 até o traço 600 tem-se traços de ambos os modelos em LS_3, indicando uma região de transição de modelo de processo. E finalmente, tem-se do traço 601 até o traço 900 apenas traços do Modelo 2.

Figura 40: Probabilidade de cada modelo e adição de traços no Modelo 1 e Modelo 2 ao longo do LS_{18}.



Fonte: O autor (2024)

O processo de validação foi definido na Subseção 3.2.3 e será aplicado, a seguir, em um dos logs gerados (LSs) na **Base Gerada 2**. Portanto, deve-se ressaltar novamente: no período de transição (traço 301 até 600), o modelo gerado combina o Modelo 1 com o Modelo 2. No caso em específico, o Modelo 2 é o Modelo 1 com uma atividade adicionada e, dessa forma, no período de transição terá traços que vão pular a atividade adicionada e traços que vão passar por ela. Isso foi confirmado ao submeter o log ao IPDD *framework fixed* com janela de 300 traços apenas para que fosse possível dividir o log em três e observar os modelos de processos gerados pelos mineradores. A Figura 41 mostra os modelos obtidos no IPDD e, no período de transição, observa-se o comportamento esperado por traços do Modelo 1 e do Modelo 2 (cf. Figura 40). Os modelos descobertos dos logs LS2_gradual, LS5_gradual, LS9_ gradual,

LS13_gradual e LS17_gradual estão representados no **APÊNDICE B- Modelos extraídos dos logs com transição gradual da Base Gerada 2**.

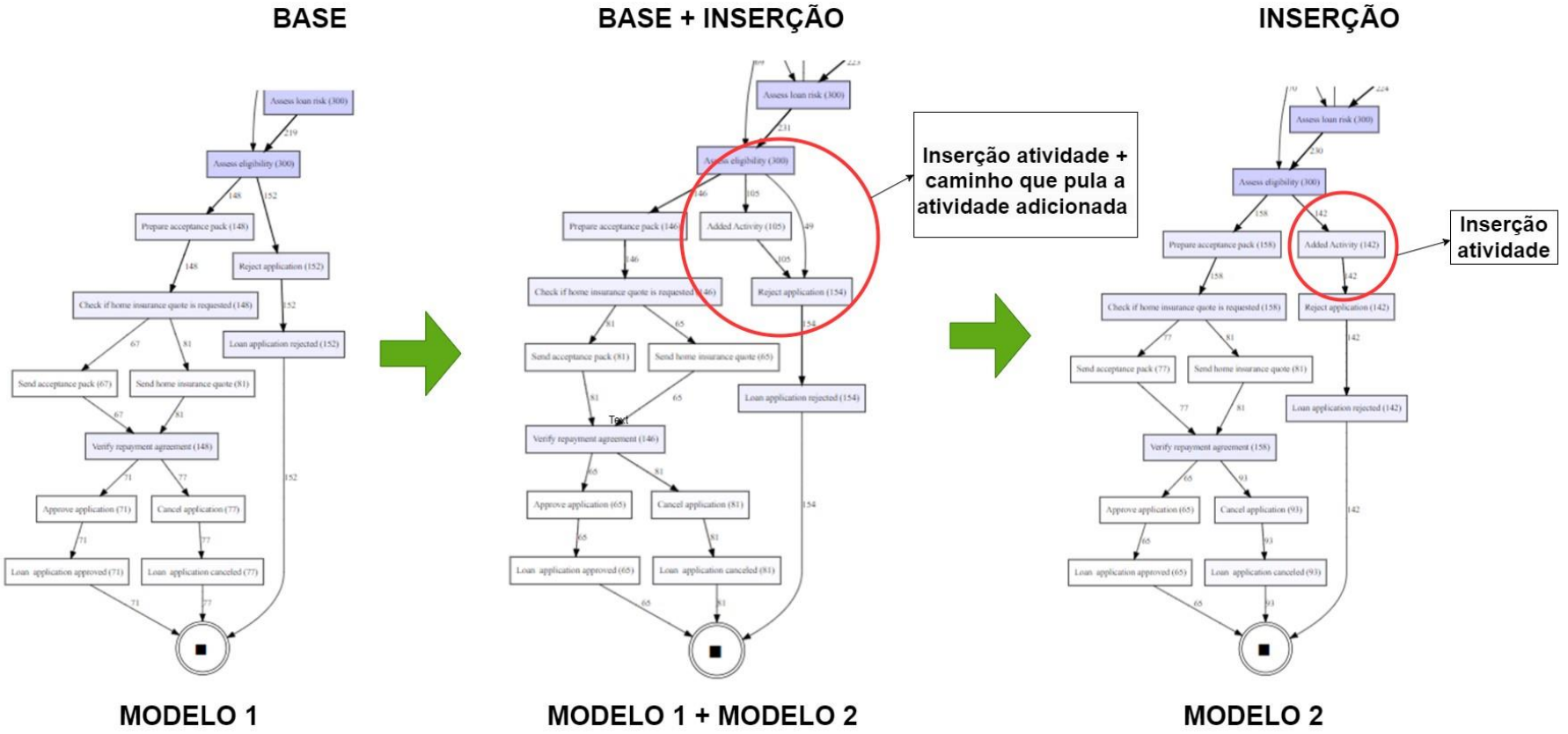
As configurações para a validação foram as seguintes em termos de ferramentas e parâmetros: Apromore- *ProDrift fixed* (*ddm runs*), IPDD *framework fixed* e o VDD *system* todos com janela fixa variando de 50, 75, 100, 125, 150, 162, 200 traços reportados no **APÊNDICE C- *F-score* validação dos logs com transição abrupta da Base Gerada 2**. O cálculo do *F-score* foi feito usando o *F-score* reativo (cf. Figura 25) — tarefa automatizada por um programa previamente desenvolvido[43] — e tolerância de erro igual ao tamanho da janela. Adicionalmente, a validação dos *drifts* recorrentes e incrementais foram feitos por meio da extração manual dos modelos e na análise visual constatou-se que estava conforme a parametrização de entrada do **PDG** (cf. Tabela 20). O **APÊNDICE D- Evolução dos modelos para a validação da evolução dos modelos no drifts incrementais e recorrentes** apresenta a análise do LS3_INCREMENTAL e LS4_RECORRENTE validando a geração de *drifts* incrementais e recorrentes.

Percebeu-se que os detectores de *drifts* — Apromore, IPDD e o VDD *system* — foram capazes de identificar *drifts* dentro da janela de tolerância, uma vez que no mínimo uma das configurações em todos os logs de eventos com transições abruptas obteve *F-score* igual a 1.

Esses resultados corroboram com a validação dos logs de eventos, feita sobre a **Base Gerada 1 (**reproduzida**)**, na medida que eles detectam os *drifts* que foram inseridos/criados artificialmente. Vale citar que a variação observada se deve as particularidades dos detectores.

[43] https://github.com/caioraduy/F-score-calculating.git

Figura 41: Divisão do LS_{18} em 3 *sub-logs* para extração dos modelos.



Fonte: O autor (2

4.3 Considerações do Capítulo

Nos 17 logs replicados/reproduzidos por 10 vezes (**Base Gerada 1 (**reproduzida**)**) observou-se que em 16 deles os resultados foram significativamente semelhantes em pelo menos 8 das 10 réplicas. Isso evidencia a capacidade do **PDG** em gerar logs de eventos de forma consistente. Posteriormente, constatou-se que a variação na frequência das atividades ocasiona mudanças no *mean delay*, o que, ao calcular o *F-score* com uma margem de erro tolerável (reativo), influencia diretamente o valor do *F-score*. Além disso, procedeu-se à validação dos logs gerados, incluindo a verificação dos logs com transições abruptas e graduais. Conclui-se que as bases geradas estão de acordo com os parâmetros de entrada do usuário e com isso a capacidade de geração de logs de eventos com *drift* foi certificada.

O próximo capítulo apresenta a comparação das ferramentas utilizando o protocolo de testes delineado na Seção 3.3.

5 COMPARAÇÃO DAS FERRAMENTAS DE DETECÇÃO DE *DRIFT*

Aplicação das ferramentas da literatura sobre o protocolo proposto de testes de análise estatística visa, *de um lado*, ilustrar a sua aplicação propriamente dita, e *de outro lado*, obter um referencial de comparação sobre o comportamento das ferramentas *vis-à-vis* a um conjunto de logs de eventos bem-conhecido. Como dito anteriormente, os conjuntos utilizados nessa comparação foram os seguintes:

- **Base Existente 1**: 18 logs de eventos extraídos da **base de logs de eventos artificias existente 3** (**BLA3**) apresentada em Maaradji et al. (2015), uma vez que o padrão *fr* foi reincluído para comparação dos detectores de *drift*;
- **Base Existente 2**: 15 logs de eventos extraídos da **base de logs de eventos artificias existente 4** (**BLA4**) apresentada em Ceravolo et al. (2020);
- **Base Gerada 2**: 15 logs de eventos com diferentes tipos de drifts gerados pelo **PDG**.

É importante destacar que os seis logs de eventos com transições graduais foram removidos da **Base Gerada 2** para permitir uma comparação adequada entre as ferramentas. Essa exclusão se deve ao fato de que os detectores são capazes apenas de identificar *drifts* com transições abruptas. Ademais, é relevante salientar que os logs com *drifts* incrementais e recorrentes podem ser considerados na comparação, uma vez que também apresentam transições abruptas. No entanto, é necessário frisar que a comparação não leva em consideração a classificação dos tipos de *drifts*, mas apenas a posição dos pontos de *drift* detectados. Assim, a presença de padrões de *drifts* recorrentes/incrementais não interfere na detecção da posição do *drift* com transição abrupta.

5.1 Seleção e parametrização dos detectores de *drifts*

A descrição dos parâmetros ao longo deste item está documentada na Tabela 21, onde cada ferramenta é associada a seus respectivos parâmetros. Uma parcela do experimento em análise foi conduzida utilizando o Apromore, o qual foi configurado para operar tanto com janelas fixas quanto adaptativas. No modo adaptativo, foi empregada a estratégia ADWIN com

a mesma variação na janela inicial, a fim de investigar se essa abordagem afeta os resultados do *F-score* para a execução da técnica *ddm runs*.

O mesmo experimento foi replicado usando o VDD *system*, utilizando o *slide size* (janela deslizante) como metade da janela — cada janela deslizante se sobrepõe à metade do seu tamanho com a janela anterior — e a opção "*driftAll*" definida como "*True*", o que indica que a análise é realizada em todo o log de eventos. Caso a opção "*driftAll*" não fosse informada, a análise de mudança seria feita por *clusters*. Os experimentos foram conduzidos utilizando diferentes janelas fixas.

No IPDD, o experimento foi conduzido com as opções "*Approach Fixed*" no IPDD *Framework fixed* e "*Approach Adaptive*" no *Adaptive* IPDD *trace by trace*. Os experimentos foram realizados com tamanhos de janelas variando no intervalo de 32 a 200 traços.

O experimento foi também parcialmente executado com *Concept Drift plugin* do ProM, utilizando apenas janelamento adaptativo sobre os 48 logs de eventos, sendo 18 logs da **Base Existente 1,** 15 logs da **Base Existente 2** e 15 da **Base Gerada 2**. A única situação em que se obteve sucesso com ProM-*Concept Drift* foi a configuração especificada na Tabela 21.

Deve-se notar que a saída de texto de cada ferramenta foi disponibilizada em um repositório[44], na forma de uma planilha contendo os dados utilizados na análise comparativa. Ademais, é válido ressaltar que os parâmetros aplicados nos três conjuntos de logs de eventos foram os mesmos.

[44] https://github.com/caioraduy/F-score-calculating

Tabela 21: Parametrização por ferramenta testada.

Ferramenta	Parametrização
Apromore *ProDrift fixed*	Janelas fixas de 32, 50, 75, 100, 125, 150, 162, 200. Abordagem *ddm runs* (Análise de traços)
Apromore ProDrift *adaptive*	Janela adaptativa Janela inicial: 32, 50, 75, 100, 125, 150, 162, 200 Abordagem *ddm runs* (Análise de traços)
VDD *system fixed*	janelas fixas de 32, 50, 75, 100, 125, 150, 162, 200. Slide size: Metade da janela Opção *driftAll = True*
IPDD *Framework fixed*	janelas fixas de 32, 50, 75, 100, 125, 150, 162, 200. Análise de traços
Adaptive* IPDD *trace by trace	Janela adaptativa Janela inicial: 32, 50, 75, 100, 125, 150, 162, 200 Abordagem *"trace by trace"* Análise de traços
ProM *Concept Drift plugin*	*Log Configuration Step: Do not split* *Feature Scope Configuration Step*: Local-todas as atividades foram selecionadas *Feature type: Follows relation* *Relation Count: Numeric value* *Metric type: J-measure* *Window size*: 10 *Drift detection algorithm: Kolmogorov-Smirnov test* *Population options: ADWIN, Sudden drift Search, trace amount* *Population Minimum Size*: 50 *Population Maximum Size*: 50 *Population Maximum Size*: 500 *P-value threshold*: 0,4 *Step Size*: 1

Fonte: O autor (2024)

5.2 Análise dos resultados – comparação

Após a execução de cada experimento sobre a **Base Existente 1,** a **Base Existente 2** e a **Base Gerada 2,** os resultados foram analisados de forma gráfica e estatística. Tal análise foi feita com base no valor de *F-score*[45], calculado em função da posição da detecção do *drift* e tolerância de erro (cf. esquema da Figura 25)

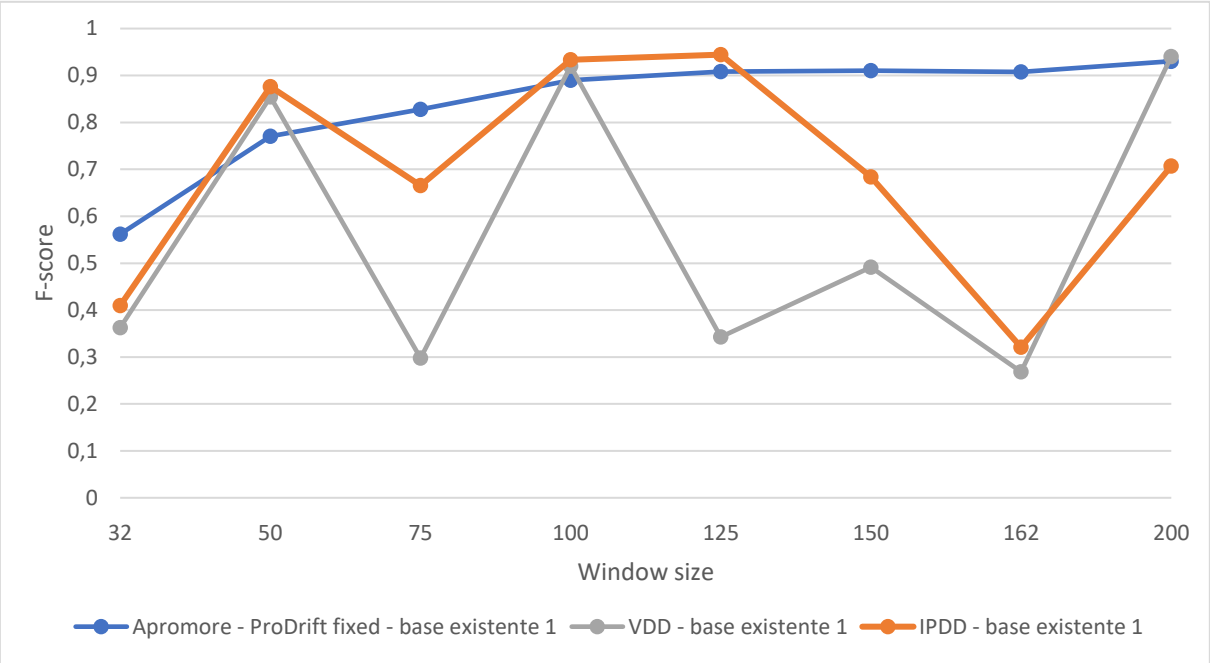
1ª Análise: influência do tamanho da janela nas abordagens com janelamento fixo

Na análise comparativa, considerando apenas os métodos/abordagens com janelamento fixo, pôde-se observar por meio das curvas da Figura 42 que o Apromore-*ProDrift fixed* obteve valores de *f-score* próximos de 1, começando em 0,55 na janela 32 e aumentando até chegar em 0,91 na janela 200 na **Base Existente 1**. Por sua vez, o VDD *system* apresentou a maior variação, com a média máxima em 0,94 e a menor média em 0,27. Já o IPDD *Framework fixed* alcançou o maior *f-score* (0,94), embora também tenha exibido instabilidade em relação ao tamanho inicial da janela.

Na **Base Existente 2** (Figura 43), o desempenho do Apromore-*ProDrift fixed* manteve o mesmo padrão de crescimento do *f-score*, alcançando um valor máximo de 0,87 para as janelas indicados. O VDD *system*, por sua vez, não apresentou nenhum resultado superior ao Apromore e obteve um *f-score* igual a zero para as janelas 150 e 162. Já o IPDD *Framework fixed* apresentou a melhor média de *f-score* (0,87) para três tamanhos de janelas (50, 100 e 125), mas para a janela de tamanho 162 o *f-score* foi igual a zero.

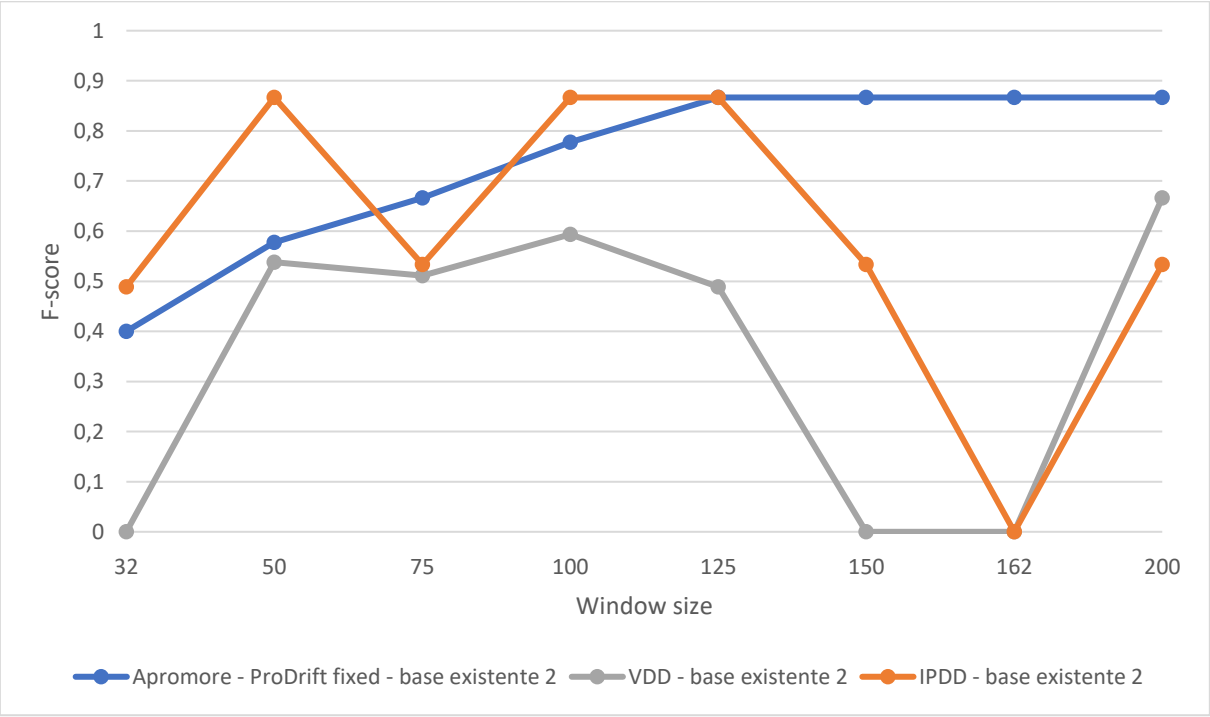
[45] https://www.kaggle.com/datasets/caioraduy/comparaorealizadanocaptulo5

Figura 42: *F-score* médio de acordo com variação do tamanho da janela inicial nos métodos de janelamento fixo para a Base Existente 1**.**



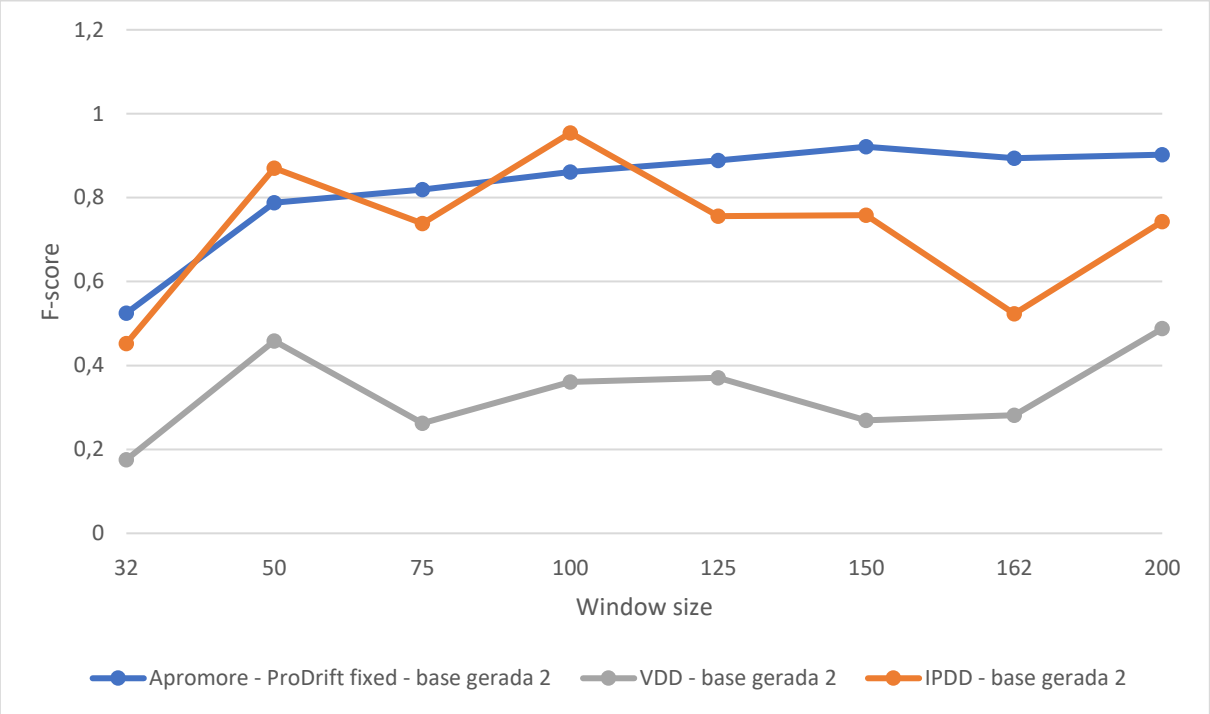
Fonte: O autor (2024)

Figura 43: *F-score* médio de acordo com variação do tamanho da janela inicial nos métodos de janelamento fixo para a Base Existente 2.



Fonte: O autor (2024)

Figura 44: F-score médio de acordo com variação do tamanho da janela inicial nos métodos de janelamento fixo para a Base Gerada 2.



Fonte: O autor (2024)

Na segunda base de dados gerada (**Base Gerada 2**), a análise realizada pelo Apromore-*ProDrift fixed* confirmou os seguintes resultados: houve uma melhora significativa nos resultados ao utilizar uma janela maior (cf. Figura 44) e observou-se uma menor oscilação durante a variação da janela (como ilustrado nas Figura 42, Figura 43 e Figura 44).

Para aprimorar a análise, além da avaliação visual dos resultados, foi conduzido o teste de *Friedman*, utilizando um nível de significância de $\alpha = 0{,}05$. Este teste visa determinar se as discrepâncias nos valores de *f-score* são estatisticamente relevantes e se o tamanho da janela tem impacto na precisão de cada ferramenta. Os resultados deste teste, exibidos na Tabela 22, revelam diferenças estatisticamente significativas na precisão das ferramentas em ambos os conjuntos de logs de eventos, levando em conta a variação no tamanho da janela de análise.

Tabela 22: Valores do Teste de *Friedman* para as abordagens com janelamento fixo.

Ferramenta	Base Existente 1 (N=18)	Base Existente 2 (N=15)	Base Gerada 2 (N=15)
Apromore–*ProDrift*	$[\chi^2(7) = 67.1;\ p < 0.001]$	$[\chi^2(7) = 33.1;\ p < 0.001]$	$[\chi^2(7) = 34.779;\ p < 0.001]$
VDD *system*	$[\chi^2(7) = 100.9;\ p < 0.001]$	$[\chi^2(7) = 85.5;\ p < 0.001]$	$[\chi^2(7) = 28.566;\ p < 0.001]$
IPDD *Fixed*	$[\chi^2(7) = 101.0;\ p < 0.001]$	$[\chi^2(7) = 88.5;\ p < 0.001]$	$[\chi^2(7) = 69.819;\ p < 0.001]$

Fonte: O autor (2024)

Na Tabela 22, $\chi^2(7)$ representa a estatística *qui-quadrado* de *Friedman* calculada para 8 elementos pareados e grau de liberdade é igual a 7. O *p representa o p-valor* que determina se existe ou não diferença com base nas hipóteses, por definição (SIEGEL; CASTELLAN JR, 2006):

Deve-se salientar que a hipótese nula é rejeita quando *p-valor* < 0,05. Para a aplicação nesse caso do teste de *Friedman*, tem-se:

H0: O tamanho da janela fixa não influencia o *f-score.*

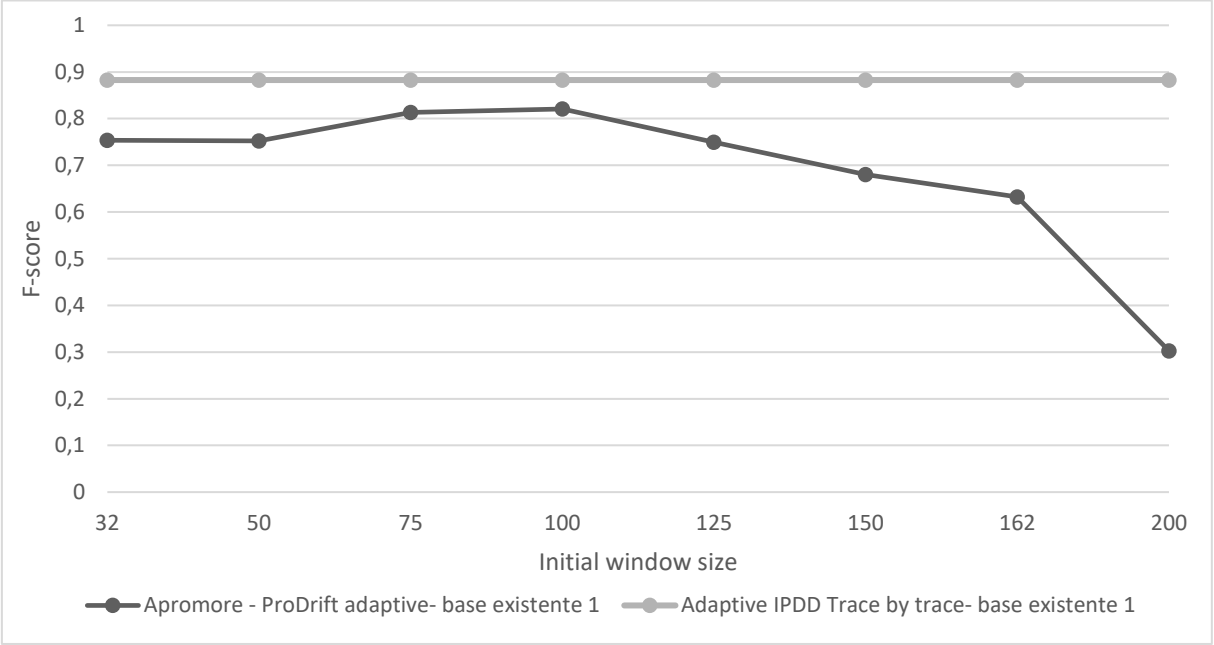
H1: O tamanho da janela fixa influencia o *f-score.*

Se a hipótese nula for rejeitada, pode-se explorar estatisticamente com outras abordagens estatísticas para explorar em quais condições as medianas são diferentes. Observa-se, na Tabela 22, que em todas as aplicações do teste de *Friedman*, a **H0** foi rejeitada.

2ª Análise: influência do tamanho da janela nas abordagens adaptativas

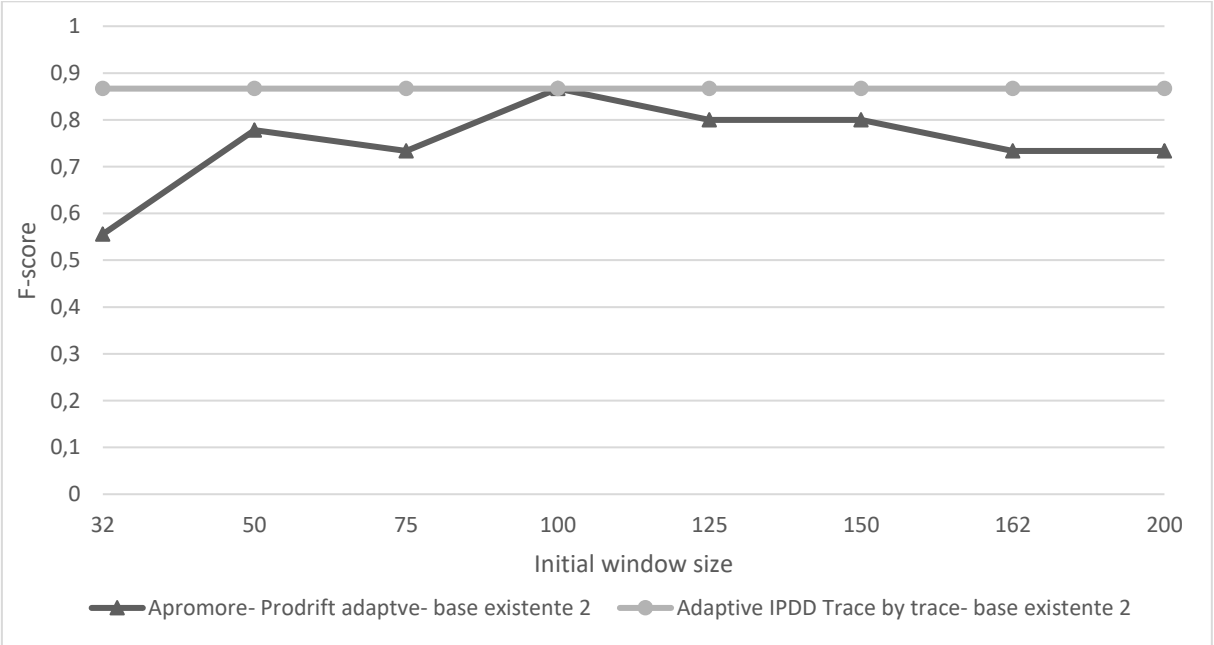
Na segunda análise, foi avaliado o impacto do tamanho da janela inicial nas abordagens com janelamento adaptativo. O IPDD *Framework trace by trace*, usando a abordagem adaptativa, alcançou valores médios consistentes ao longo das janelas para os três conjuntos de logs de eventos (cf. Figura 45, Figura 46 e Figura 47). Entretanto, ao analisar a **Base Gerada 2**, o teste de Friedman com 5% de significância indicou uma diferença estatisticamente significativa. Por outro lado, o Apromore-*ProDrift adaptive* apresentou variação no *f-score* e resultados inferiores aos obtidos com o *Adaptive* IPDD *trace by trace*. Os testes de *Friedman* (Tabela 23) confirmaram que a diferença observada para a **Base Existente 1** e a **Base Gerada 2** com o Apromore-*ProDrift adaptive* foi estatisticamente significativa, enquanto que para a **Base Existente 2** não apresentou diferença significativa (cf. Tabela 23).

Figura 45: *F-score* médio de acordo com variação do tamanho janela inicial no método adaptativo para a Base Existente 1.



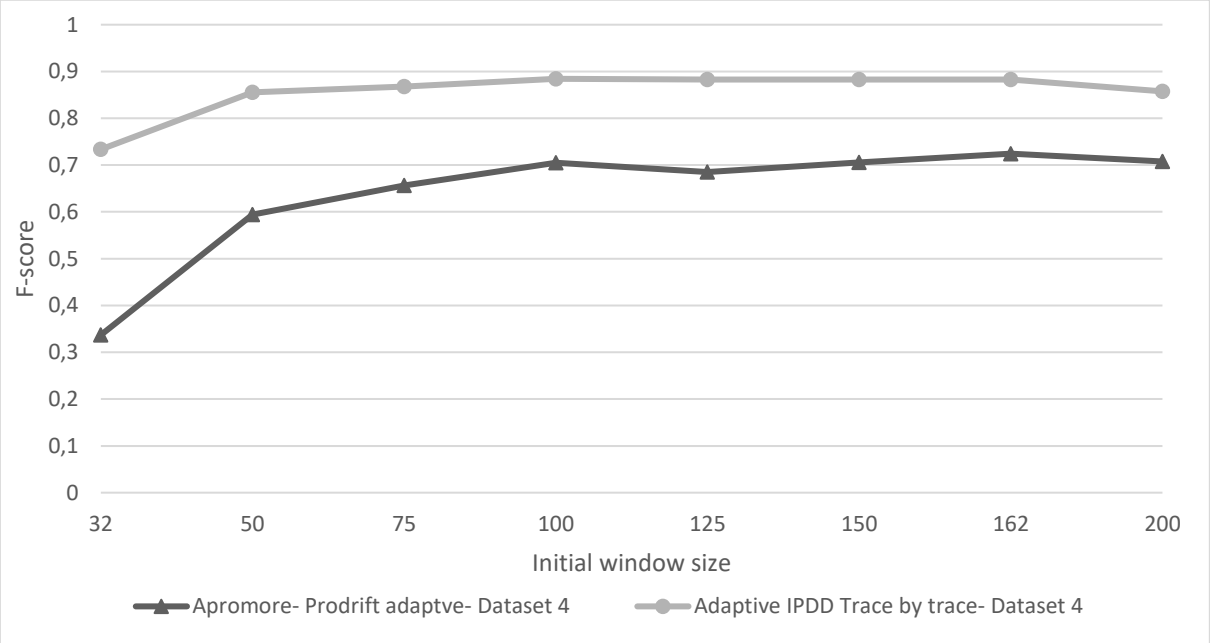
Fonte: O autor (2024)

Figura 46: F-score médio de acordo com variação do tamanho janela inicial no método adaptativo para a Base Existente 2.



Fonte: O autor (2024)

Figura 47: *F-score* médio de acordo com variação do tamanho janela inicial no método adaptativo para a Base Gerada 2.



Fonte: O autor (2024)

Tabela 23: Valores do teste de *Friedman* para as abordagens com janelamento adaptativo.

	Base Existente 1 (N=18)	**Base Existente 2 (N=15)**	**Base Gerada 2 (N=15)**
Apromore–*ProDrift Adaptive*	$[\chi^2(7) = 42.9;\ p < 0.001]$	$[\chi^2(7) = 11.3;\ p = 0.127\ ]$	$[\chi^2(7) = 31.1;\ p < 0.001\ ]$
Adaptive* IPDD *trace by trace	$[\chi^2(7) = NaN;\ p = NaN]$	$[\chi^2(7) = NaN;\ p = NaN]$	$[\chi^2(7) = 30.4;\ p < 0.001\ ]$

Fonte: O autor (2024)

A Tabela 23 mostra que apenas os testes realizados na **Base Existente 2,** aplicando o Apromore-*ProDrift* no janelamento adaptativo, rejeitaram a hipótese nula (*p-valor* > 0,05). O *Adaptive* IPDD *trace by trace e* no janelamento adaptativo obteve *NaN* como resultado do teste. Isso aconteceu porque, nas 8 amostras, os valores de *f-score* para cada log foram iguais. Os empates geram um problema no cálculo da estatística do teste de *Friedman*. Assim, assume-se que, para o *Adaptive* IPDD *trace by trace*, não existe diferença entre as amostras na **Base Existente 1** e na **Base Existente 2**.

3ª Análise: comparação da acurácia

A comparação da acurácia foi feita com os melhores resultados obtidas para cada um dos detectores de *drifts* em questão. A Tabela 24 mostra as médias dos melhores resultados

obtidos pelas ferramentas sobre a **Base Existente 1** e a **Base Existente 2**. O teste de *Friedman* (cf. Tabela 25) apontou a existência de diferença significativa em ambos os conjuntos de logs de eventos. Complementarmente, foi feito o teste de *Nemenyi* para explorar a diferença entre as abordagens.

Tabela 24: Melhores médias de cada abordagens para os conjuntos de logs de eventos testados.

Conjunto de logs de eventos	**Apromore-*ProDrift fixed***	**VDD *system***	**Apromore-*ProDrift adaptive***	**ProM-*Concept Drift adaptive***	**IPDD *framework fixed***	**Adaptive IPDD *trace by trace***
Base existente 1	0,93	0,94	0,82	0,12	0,93	0,88
Base existente 2	0,86	0,66	0,86	0,33	0,86	0,86
Base gerada 2	0,92	0,48	0,72	0,05	0,95	0,88

Fonte: O autor (2024)

Tabela 25: Resultados do teste de *Friedman* para a comparação dos melhores resultados das ferramentas.

	Base existente 1 (N=18)	**Base existente 2 (N=15)**	**Base gerada 2 (N=15)**
Abordagens	$[\chi^2(5) = 55{,}7;\ p\ <\ 0.001]$	$[\chi^2(5) = 29{,}0;\ p < 0.001\ ]$	$\chi^2(5) = 52{,}5;\ p < 0.001\ ]$

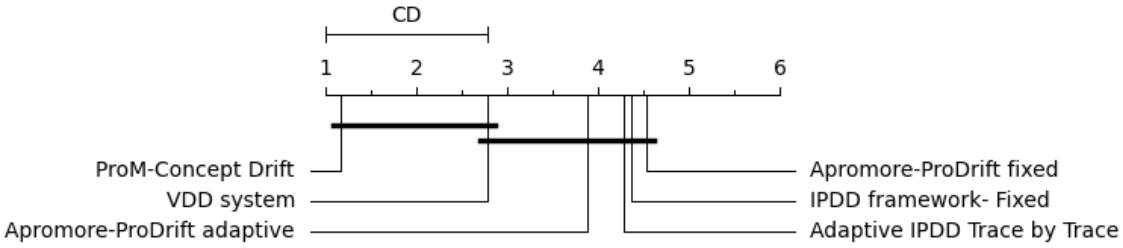
Fonte: O autor (2024)

A análise por meio do teste de *Nemenyi* para a **Base Existente 1** (cf. Figura 48) mostrou que o Apromore-*ProDrift fixed*, Apromore-*ProDrift adaptive*, *Adaptive* IPDD *trace by trace*, IPDD *framework fixed* e VDD *system* obtiveram resultados de *f-score* mais próximos de 1 e esses foram significativamente iguais entre eles. O ProM-*Concept Drift* mostrou-se significativamente igual ao VDD *system*, mas diferente face as outras abordagens.

O teste de *Nemenyi* para a **Base Existente 2** (cf. Figura 49) não revelou diferença significativa entre os grupos avaliados. Essa ausência de significância ocorre quando a diferença observada nos dados é próxima da margem de erro estabelecida pelo teste. Por essa razão, o teste de *Friedman* é inicialmente preferido, dado seu maior poder estatístico, o que o torna mais confiável que o teste de *Nemenyi*. Por outro lado, ao analisar a **Base Gerada 2** através do teste de Nemenyi (cf. Figura 50) observou-se que:

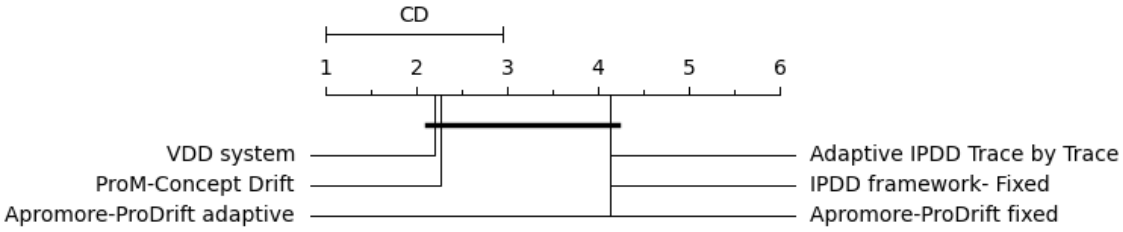
- Apromore-*ProDrift fixed*, Apromore-*ProDrift adaptive*, Adaptive IPDD *trace by trace*, IPDD *framework fixed* são significativamente iguais;
- o VDD *system* e o Apromore-*ProDrift adaptive* são significativamente iguais;
- o VDD *system* e o ProM- *Concept drift* são significativamente iguais.

Figura 48: Teste de *Nemenyi* para comparar os melhores resultados obtidos das ferramentas submetidas a Base Existente 1.



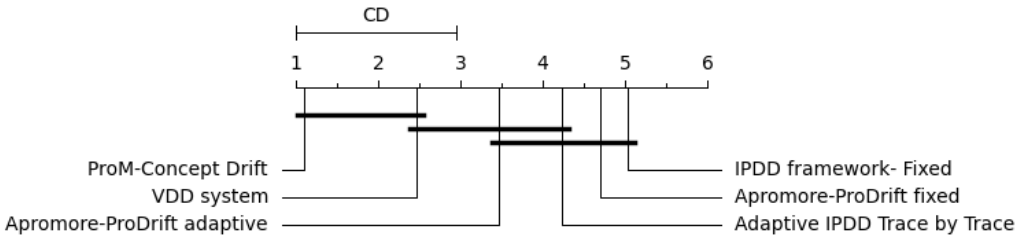
Fonte: O autor (2024)

Figura 49: Teste de *Nemenyi* para comparar os melhores resultados obtidos das ferramentas submetidas a Base Existente 2.



Fonte: O autor (2024)

Figura 50: Teste de *Nemenyi* para comparar os melhores resultados obtidos das ferramentas submetidas a Base Gerada 2.



Fonte: O autor (2024)

Nas figuras (Figura 48, Figura 49 e Figura 50), as amostras que são significativamente iguais estão ligadas entre si por uma linha em **negrito**. Essa linha é determinada pela distância crítica (CD) descrita na Subseção 3.3.2. Ela representa a menor diferença que dois grupos precisam ter para serem estatisticamente diferentes.

5.3 Considerações do capítulo

A análise descrita até o momento visou examinar o comportamento de diferentes detectores de *drifts* em processos e mostrar a aplicação do protocolo de teste descrito na Seção 3.3. A primeira análise revelou que em todas as abordagens de janelamento fixo, o tamanho da janela influenciou no *f-score* para a **Base Existente 1**, **Base Existente 2** e **Base Gerada 2**. Na segunda análise o Apromore-*ProDrift adaptive* revelou que o *f-score* é influenciado pela janela inicial no **Base Existente 1** e na **Base Gerada 2**. Para o *Adaptive* IPDD *trace by trace*, encontrou-se apenas diferença significativa entre as amostras na **Base Gerada 2.** Na terceira análise, Apromore-*ProDrift fixed*, Apromore-*ProDrift adaptive*, *Adaptive* IPDD *trace by trace* e IPDD *framework fixed* obtiveram os melhores resultados para o **Base Existente 1** e na **Base Gerada 2** sendo significativamente iguais. No entanto, na **Base Existente 2**, a investigação da diferença com o teste de *Nemenyi* não encontrou diferença estatisticamente significativas entre os pares.

6 CONLUSÕES E TRABALHOS FUTUROS

A revisão da literatura proporcionou uma análise dos logs de eventos com *drifts* e dos geradores de logs de eventos com *drifts* já existentes. Esse exame permitiu a identificação e o mapeamento das principais abordagens, fornecendo um embasamento teórico para a concepção do método computacional desenvolvido nesta pesquisa, denominado **PDG**. A partir desse embasamento, foi possível situar o **PDG** no contexto atual de pesquisa, destacando sua contribuição e sua relevância para o avanço do conhecimento nesta área. Além disso, a revisão da literatura proporcionou percepções importantes sobre lacunas e desafios existentes, orientando o desenvolvimento e a validação do método proposto. Deste modo, a análise crítica realizada neste estudo não apenas consolida o estado atual do conhecimento, mas também oferece uma base para futuras investigações e desenvolvimentos nesse campo, contribuindo de forma significativa para a evolução e aprimoramento das técnicas de geração e análise de logs de eventos com *drifts*.

De forma mais pontual, **PDG** permite gerar logs de eventos com diferentes tamanhos de logs de eventos, intervalos variáveis entre *drifts*, tipos diferentes de *drifts*, número de *drifts* variáveis e ruído. Vale citar que o estudo das funções de probabilidades permitiu a implementação de *drifts* graduais com decaimento linear ou exponencial. Além disso, foi definido um protocolo de experimentos da comparação de ferramentas de detecção de *drift* publicado na conferência ICAISC2022 (QUALIS A2).

O **PDG** foi validado em duas análises. Na primeira validação (Seção 4.1), permitiu replicar os logs de eventos de uma base de dados consolidada na literatura. Sobre tais réplicas foram aplicados detectores de *drifts* e calculados os seus *f-score*. E cada réplica/log foi comparada com o log de eventos original—ou da literatura—, e avaliado estatisticamente se eles eram significativamente iguais. Os resultados obtidos revelaram que os logs replicados tiveram 80% de confiança mínima (no mínimo 8 vezes das 10 vezes em que ele foi replicado, o teste estatístico indicou que os resultados do *f-score* eram significativamente iguais em 16 dos 17 logs). Isso acontece porque o simulador do PM4PY gera traços com frequências diferentes e influencia diretamente na posição do *drift* detectado pelo gerador. Uma vez que o *f-score* é calculado com uma janela de tolerância igual ao tamanho da janela inicial, qualquer diferença na posição do *drift* detectado gera um grande impacto no *f-score* .

Na segunda validação (Seção 4.2), foram gerados 21 logs de eventos com diversas configurações, dando forma ao **Base Gerada 2**, mostrando a capacidade de geração do **PDG**. Percebeu-se que os detectores de *drifts* — Apromore, IPDD e o VDD *system* — foram capazes de identificar *drifts* dentro da janela de tolerância, uma vez que no mínimo uma das configurações em todos os logs de eventos com transições abruptas obteve *f-score* igual a 1. Além disso, os logs de eventos graduais foram validados com a divisão dos logs de eventos em *sub-logs* e a análise visual dos modelos. Assim, com essas duas análises o objetivo **OE1** foi atingido.

A execução do protocolo de testes em dois conjuntos de logs de eventos — **Base Existente** 1 e **Base Existente 2** —, seguida pela sua implementação na **Base Gerada 2**, proporcionou percepções importantes sobre a eficácia das ferramentas de detecção de *drift*. Ficou claro que a precisão dos métodos com janelamento fixo é diretamente afetada pelo tamanho da janela, enquanto os métodos com janelamento adaptativo são sensíveis à escolha da janela inicial. Além disso, o excelente desempenho das ferramentas na detecção da posição dos *drifts*, especialmente em relação à **Base Gerada 2**, serve como uma validação do método **PDG** para geração de logs de eventos que apresentam diferentes tipos de *drifts*. Esta constatação reforça a confiabilidade e a robustez do método proposto. Esses resultados destacam a importância de considerar as nuances dos *drifts* ao desenvolver e avaliar ferramentas de detecção, além de ressaltar a necessidade contínua de pesquisa e desenvolvimento nesta área em constante evolução.

Ao comparar o desempenho geral das ferramentas, destacam-se o Apromore-*ProDrift fixed*, Apromore-*ProDrift adaptive*, *Adaptive* IPDD *trace by trace* e IPDD *Framework fixed*, os quais apresentaram os melhores resultados tanto para a **Base Existente 1** quanto para a **Base Gerada 2**, mostrando uma significativa igualdade entre elas. Com a comparação das ferramentas com a **bases existentes** e **gerada**, o **OE2** foi atingido e foi possível observar uma aplicação prática da utilização da **Base Gerada 2.**

TRABALHOS FUTUROS

Perante a constatação de que atrasos na detecção de um *drift* influenciam diretamente nas comparações quando se utiliza o *f-score* em conjunto erro de tolerância (reativo), conforme Figura 25, sugere-se a replicação dos experimentos utilizando o *f-score* com *mean delay*, conforme, Figura 23 proposto em Sato (2022). Na Subseção 4.1.2, a investigação das diferenças

entre a **Base Gerada 1** e a **Base Existente 1** é a motivação da utilização do *f-score* com *mean delay*. No entanto, deve-se padronizar a utilização da métrica. Por isso, como trabalhos futuros, sugere-se a replicação dos seguintes experimentos:

- Validação da **Base Gerada 1** utilizando *o f-score* com *mean delay*;
- Validação da **Base Gerada 2** utilizando o *f-score* com *mean delay*;
- Comparação das ferramentas de detecção de *drift* utilizando o *f-score* com *mean delay*.

Essas investigações fornecerão percepções importantes sobre a eficácia e a robustez das abordagens propostas, contribuindo para aprimorar a compreensão e a aplicabilidade das técnicas de detecção de *drift* em contextos de análise de dados.

7 REFERÊNCIAS

AALST, W. VAN DER. **Process mining: Data science in action**. [s.l.] Springer, 2016.

ADAMS, J. N. et al. **A Framework for Explainable Concept Drift Detection in Process Mining**. 19th International Conference, BPM 2021 - Lecture Notes in Computer Science. **Anais**...Rome, Italy: Springer Science and Business Media Deutschland GmbH, 2021.

ADAMS, J. N. et al. Explainable concept drift in process mining. **Information Systems**, v. 114, p. 102177, 2023a.

ADAMS, J. N. et al. An Experimental Evaluation of Process Concept Drift Detection. **Proceedings of the VLDB Endowment**, v. 16, n. 8, p. 1856–1869, 2023b.

BAIER, L.; REIMOLD, J.; KUHL, N. Handling Concept Drift for Predictions in Business Process Mining. **Proceedings - 2020 IEEE 22nd Conference on Business Informatics, CBI 2020**, v. 1, p. 76–83, 2020.

BARBON JUNIOR, S. et al. **A Framework for Human-in-the-loop Monitoring of Concept-drift Detection in Event Log Stream**. WWW '18: Companion Proceedings of the The Web Conference 2018. **Anais**...Lyon, France: Association for Computing Machinery (ACM), 2018.

BOLT, A.; LEONI, M. DE; AALST, W. M. P. VAN DER. Scientific workflows for process mining: building blocks, scenarios, and implementation. **International Journal on Software Tools for Technology Transfer**, v. 18, n. 6, p. 607–628, 2016.

BOSE, R. P. J. C. et al. **Handling concept drift in process mining**. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). **Anais**...Springer, Berlin, Heidelberg, 2011. Disponível em: <http://link.springer.com/10.1007/978-3-642-21640-4_30>. Acesso em: 16 ago. 2019

BOSE, R. P. J. C. et al. Dealing with concept drifts in process mining. **IEEE Transactions on Neural Networks and Learning Systems**, v. 25, n. 1, p. 154–171, 2014.

BROCKHOFF, T.; UYSAL, M. S.; AALST, W. M. P. VAN DER. **Time-aware Concept Drift Detection Using the Earth Mover's Distance**. 2nd International Conference on Process Mining (ICPM). **Anais**...Padua, Italy: IEEE, out. 2020. Disponível em: <https://ieeexplore.ieee.org/document/9230247/>. Acesso em: 4 nov. 2020

BURATTIN, A. et al. Online Discovery of Declarative Process Models from Event Streams. **IEEE Transactions on Services Computing**, v. 8, n. 6, p. 833–846, 2015.

BURATTIN, A. PLG2: Multiperspective process randomization with online and offline simulations. **CEUR Workshop Proceedings**, v. 1789, p. 1–6, 2016.

BURATTIN, A.; SPERDUTI, A. PLG: A framework for the generation of business process models and their execution logs. **Lecture Notes in Business Information Processing**, v. 66 LNBIP, p. 214–219, 2011.

CERAVOLO, P. et al. Evaluation Goals for Online Process Mining: a Concept Drift Perspective. **IEEE Transactions on Services Computing**, 2020.

CHE, H.; MACHU, Q.; ZHOU, Y. **Time drift detection in process mining**. 2016.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, v. 7, p. 1–30, 2006.

GALLEGO-FONTENLA, V.; VIDAL, J. C.; LAMA, M. A Conformance Checking-Based Approach for Sudden Drift Detection in Business Processes. **IEEE Transactions on Services Computing**, v. 16, n. 1, p. 13–26, 2023.

GRIMM, J.; KRAUS, A.; AA, H. VAN DER. CDLG: A Tool for the Generation of Event Logs with Concept Drifts. **CEUR Workshop Proceedings**, v. 3216, p. 92–96, 2022.

GRÜGER, J. et al. SAMPLE: A Semantic Approach for Multi-perspective Event Log Generation. **Lecture Notes in Business Information Processing**, v. 468 LNBIP, p. 328–340, 2023.

GUAN, W. et al. AIMED: An automatic and incremental approach for business process model repair under concept drift. **Information Systems**, v. 119, n. September, p. 102285, 2023.

HANGA, K. M.; KOVALCHUK, Y.; GABER, M. M. PGraphD*: Methods for Drift Detection and Localisation Using Deep Learning Modelling of Business Processes. **Entropy**, v. 24, n. 7, p. 1–31, 2022.

HASSANI, M. **Concept drift detection of event streams using an adaptive window**. Proceedings - European Council for Modelling and Simulation, ECMS. **Anais**...Napoli, Italy: European Council for Modelling and Simulation, 2019.

HERBET, T.; MANGLER, J.; RINDERLE-MA, S. **Generating Reliable Process Event Streams and Time Series Data based on Neural Networks**. BPMDS 2021, EMMSAD 2021: Enterprise, Business-Process and Information Systems Modeling. **Anais**...2021.

HMAMI, A.; SBAI, H.; FREDJ, M. Enhancing change mining from a collection of event logs: Merging and Filtering approaches. **Journal of Physics: Conference Series**, v. 1743, n. 1, 2021.

HMANI, A.; SBAI, H.; FREDJ, M. Handling Sudden and Recurrent Changes in Business Process Variability: Change Mining based Approach. **International Journal of Advanced Computer Science and Applications**, v. 12, n. 4, p. 632–640, 2021.

HOMPES, B. et al. **Detecting Changes in Process Behavior Using Comparative Case Clustering**. Data-Driven Process Discovery and Analysis (SIMPDA 2015). **Anais**...Vienna, Austria: Springer, 2017. Disponível em: <http://link.springer.com/10.1007/978-3-319-53435-0>

HOMPES, B. F. A. et al. Detecting change in processes using comparative trace clustering. **CEUR Workshop Proceedings**, v. 1527, n. November, p. 95–108, 2015.

HUANG, T. et al. A fog computing based concept drift adaptive process mining framework for mobile APPs. **Future Generation Computer Systems**, v. 89, p. 670–684, 2018.

HUETE, J.; QAHTAN, A. A.; HASSANI, M. PrefixCDD: Effective Online Concept Drift Detection over Event Streams using Prefix Trees. **Proceedings - International Computer Software and Applications Conference**, v. 2023- June, p. 328–333, 2023.

JENSEN, K. A brief introduction to coloured petri nets. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 1217, p. 203–208, 1997.

KUMAR, M. V. M.; THOMAS, L.; BASAVA, A. **Concept drifts detection and localisation in process mining**. International Conference on Management, Economics and Business Information. **Anais**...Tokyo, Japan: 2016.

LI, T. et al. **Unraveling Process Evolution by Handling Concept Drifts in Process Mining**. 2017 IEEE International Conference on Services Computing (SCC). **Anais**...2017.

LIN, L. et al. LCDD: Detecting Business Process Drifts Based on Local Completeness. **IEEE Transactions on Services Computing**, v. 14, n. 8, p. 1–1, 2020.

LIN, L. et al. **Process Drift Detection in Event Logs with Graph Convolutional Networks**. DASFAA 2023: Database Systems for Advanced Applications. **Anais**...2023.

LIU, N.; HUANG, J.; CUI, L. **A Framework for Online Process Concept Drift Detection from Event Streams**. 2018 IEEE International Conference on Services Computing (SCC). **Anais**...San Francisco, CA, USA: IEEE, 2018.

LU, Y.; CHEN, Q.; POON, S. **A Robust and Accurate Approach to Detect Process Drifts from Event Streams**. BPM 2021: Business Process Management. **Anais**...2021a.

LU, Y.; CHEN, Q.; POON, S. Detecting and Understanding Branching Frequency Changes in Process Models. **Lecture Notes in Business Information Processing**, v. 421, p. 39–46, 2021b.

LUENGO, D.; SEPÚLVEDA, M. **Applying Clustering in Process Mining to Find Different Versions of a Business Process that Changes over Time**. (F. Daniel, K. Barkaoui, S. Dustdar, Eds.)Business Process Management Workshops. **Anais**...Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. Disponível em: <http://link.springer.com/10.1007/978-3-642-28108-2_15>

MAARADJI, A. et al. **Fast and Accurate Business Process Drift Detection**. BPM 2016: Business Process Management. **Anais**...Innsbruck, Austria: Springer, Cham, 2015. Disponível em: <http://link.springer.com/10.1007/978-3-319-23063-4_27>. Acesso em: 16 ago. 2019

MAARADJI, A. et al. Detecting Sudden and Gradual Drifts in Business Processes from Execution Traces. **IEEE Transactions on Knowledge and Data Engineering**, v. 29, n. 10, p. 2140–2154, 2017.

MAGGI, F. M. et al. Online process discovery to detect concept drifts in LTL-based declarative process models. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 8185 LNCS, n. September, p. 94–111, 2013.

MANNHARDT, F. **Hospital Billing - Event Log. Version 1**. , 2017. Disponível em: <https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741>

MARTJUSHEV, J.; BOSE, R. P. J. C.; AALST, W. M. P. VAN DER. **Change Point Detection and Dealing with Gradual and Multi-Order Dynamics in Process Mining**. International Conference on Business Informatics Research. **Anais**...2015.

MEIRA NETO, A. C. et al. Towards a Transition Matrix-Based Concept Drift Approach: Experiments on the Detection Task. **International Conference on Enterprise Information Systems, ICEIS - Proceedings**, v. 2, n. Iceis, p. 361–372, 2023.

MEIRA NETO, A. C. et al. Revisiting the Transition Matrix-Based Concept Drift Approach: Improving the Detection Task Reliability Through Additional Experimentation. **SN Computer Science**, v. 5, n. 1, 2024.

OMORI, N. J. et al. **Comparing Concept Drift Detection with Process Mining Tools**. SBSI'19: Proceedings of the XV Brazilian Symposium on Information Systems. **Anais**...Aracaju, Brazil: ACM, 2019.

OSTOVAR, A. et al. **Detecting Drift from Event Streams of Unpredictable Business Processes**. Conceptual Modeling: 35th International Conference, ER 2016 (Lecture Notes in Computer Science). **Anais**...Springer, Cham, 14 nov. 2016. Disponível em: <http://link.springer.com/10.1007/978-3-319-46397-1_26>. Acesso em: 16 ago. 2019

OSTOVAR, A. et al. **Characterizing Drift from Event Streams of Business Processes**. International Conference on Advanced Information Systems Engineering CAiSE 2017: Advanced Information Systems Engineering. **Anais**...Essen, Germany: Springer, 2017. Disponível em: <http://link.springer.com/10.1007/978-3-319-59536-8_14>. Acesso em: 16 ago. 2019

OSTOVAR, A.; LEEMANS, S. J. J.; ROSA, M. LA. Robust Drift Characterization from Event Streams of Business Processes. **ACM Transactions on Knowledge Discovery from Data**, v. 14, n. 3, p. 1–57, 14 maio 2020.

POURMASOUMI, A. et al. On business process variants generation. **CEUR Workshop Proceedings**, v. 1367, p. 179–188, 2015.

POWERS, D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. p. 37–63, 2020.

RADUY, C. et al. **A Benchmark of Process Drift Detection Tools: Experimental Protocol and Statistical Analysis**. ICAISC 2022: Artificial Intelligence and Soft Computing. **Anais**...Springer, Cham, 2023. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-031-23480-4_12>. Acesso em: 13 fev. 2023

RATZER, A. V. et al. CPN tools for editing, simulating, and analysing Coloured Petri Nets. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 2679, p. 450–462, 2003.

RICHTER, F.; SEIDL, T. **TESSERACT: Time-Drifts in Event Streams Using Series of Evolving Rolling Averages of Completion Times**. International Conference on Business Process Management BPM 2017: Business Process Management. **Anais**...Barcelona, Spain: Springer, 2017. Disponível em: <http://link.springer.com/10.1007/978-3-319-65000-5_17>

RICHTER, F.; SEIDL, T. Looking into the TESSERACT: Time-drifts in event streams using series of evolving rolling averages of completion times. **Information Systems**, v. 84, p. 265–282, 2019.

SATO, D. M. V. **Concept Drift Detection in Process Models**. [s.l.] PUC-PR, 2022.

SATO, D. M. V. et al. A Survey on Concept Drift in Process Mining. **ACM Computing Surveys**, v. 54, n. 9, p. 1–38, 7 out. 2022.

SATO, D. M. V.; BARDDAL, J. P.; SCALABRIN, E. E. **Interactive Process Drift Detection Framework**. ICAISC 2021: Artificial Intelligence and Soft Computing. **Anais**...Zakopane, Poland: Springer, Cham, 20 jun. 2021. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-030-87897-9_18>. Acesso em: 23 dez. 2021

SCHLIMMER, J. C.; GRANGER JR, R. H. Beyond incremental processing: Tracking concept drift. **Proceedings of the Fifth National Conference on Artificial Intelligence**, v. 1, p. 502–507, 1986.

SEELIGER, A.; NOLLE, T.; MÜHLHÄUSER, M. **Detecting concept drift in processes using graph metrics on process graphs**. Proceedings of the 9th Conference on Subject-oriented Business Process Management, S-BPM ONE '17. **Anais**...Darmstadt: 2017.

SIEGEL, S.; CASTELLAN JR, N. J. **Estatística não-Paramétrica Para Ciências do Comportamento**. 2ª Edição ed. [s.l.] Artmed, 2006.

SOUSA, R. G. DE et al. Concept drift detection and localization in process mining: An integrated and efficient approach enabled by trace clustering. **Proceedings of the ACM Symposium on Applied Computing**, p. 364–373, 2021.

SOUSA, R. G. DE et al. Integrated detection and localization of concept drifts in process mining with batch and stream trace clustering support. **Data and Knowledge Engineering**, v. 149, n. November 2023, p. 102253, 2024.

SPENRATH, Y.; HASSANI, M. Predicting business process bottlenecks in online events streams under concept drifts. **Proceedings - European Council for Modelling and Simulation, ECMS**, v. 34, n. 1, p. 190–196, 2020.

STERTZ, F.; RINDERLE-MA, S. **Detecting and Identifying Data Drifts in Process Event Streams Based on Process Histories**. Information Systems Engineering in Responsible Information Systems. CAiSE 2019. **Anais**...Rome, Italy: Springer International Publishing, 2019. Disponível em: <http://dx.doi.org/10.1007/978-3-030-21297-1_12%0Ahttp://link.springer.com/10.1007/978-3-030-21297-1>

STOCKER, T.; ACCORSI, R. Secsy: Security-aware synthesis of process event logs. **Workshop on Enterprise Modelling and Information Systems Architectures**, p. 71–84, 2013.

TAVARES, G. M. et al. **Overlapping analytic stages in online process mining**. IEEE International Conference on Services Computing (SCC 2019). **Anais**...Milan, Italy: IEEE, 2019.

WOHLIN, C. **Guidelines for snowballing in systematic literature studies and a replication in software engineering**. Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14. **Anais**...New York, New York, USA: ACM Press, 2014. Disponível em: <http://dl.acm.org/citation.cfm?doid=2601248.2601268>

XU, J.; ZHANG, Y.; DUAN, Q. Concept drift detection and localization framework based on behavior replacement. **Applied Intelligence**, v. 53, n. 13, p. 16776–16796, 2023.

YAGHOUBI, M.; NAZARI, M. An efficient drift detection approach using data entropy in business processes. **Proceedings of 5th National Conference on Advances in Enterprise Architecture, NCAEA 2021**, n. Ncaea, p. 17–22, 2021.

YAHYA, B. N. et al. RT-PLG: Real Time Process Log Generator. **Advances in Intelligent Systems and Computing**, v. 431, p. 127–138, 2016.

YANG, L. et al. Process Duration Modelling and Concept Drift Detection for Business Process Mining. **Proceedings - 2021 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People, and Smart City Innovations, SmartWorld/ScalCom/UIC/ATC/IoP/SCI 2021**, p. 653–658, 2021.

YANG, L. et al. A multi-components approach to monitoring process structure and customer behaviour concept drift. **Expert Systems with Applications**, v. 210, n. June, p. 118533, 2022a.

YANG, L. et al. Detecting and Responding to Concept Drift in Business Processes. **Algorithms**, v. 15, n. 5, p. 1–19, 2022b.

YESHCHENKO, A. et al. **Comprehensive Process Drift Analysis with the Visual Drift Detection Tool**. CEUR Workshop Proceedings. **Anais**...Salvador, Brazil: CEUR-WS, 2019a.

YESHCHENKO, A. et al. **Comprehensive Process Drift Detection with Visual Analytics**. Conceptual Modeling. ER 2019. Lecture Notes in Computer Science. **Anais**...Salvador, Brazil: Springer, 2019b. Disponível em: <http://arxiv.org/abs/1907.06386>

YESHCHENKO, A. et al. **VDD: A Visual Drift Detection System for Process Mining**. ICPM 2020 Doctoral Consortium and Tool Demonstration Track. **Anais**...Padua, Italy: CEUR-WS.org, 2020a. Disponível em: <https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>. Acesso em: 2 nov. 2020

YESHCHENKO, A. et al. **Visualizing Business Process Evolution**. (S. Nurcan et al., Eds.)Enterprise, Business-Process and Information Systems Modeling. **Anais**...Cham: Springer International Publishing, 2020b.

YESHCHENKO, A. et al. Visual Drift Detection for Sequence Data Analysis of Business Processes. **IEEE Transactions on Visualization and Computer Graphics**, p. 1–1, 2021.

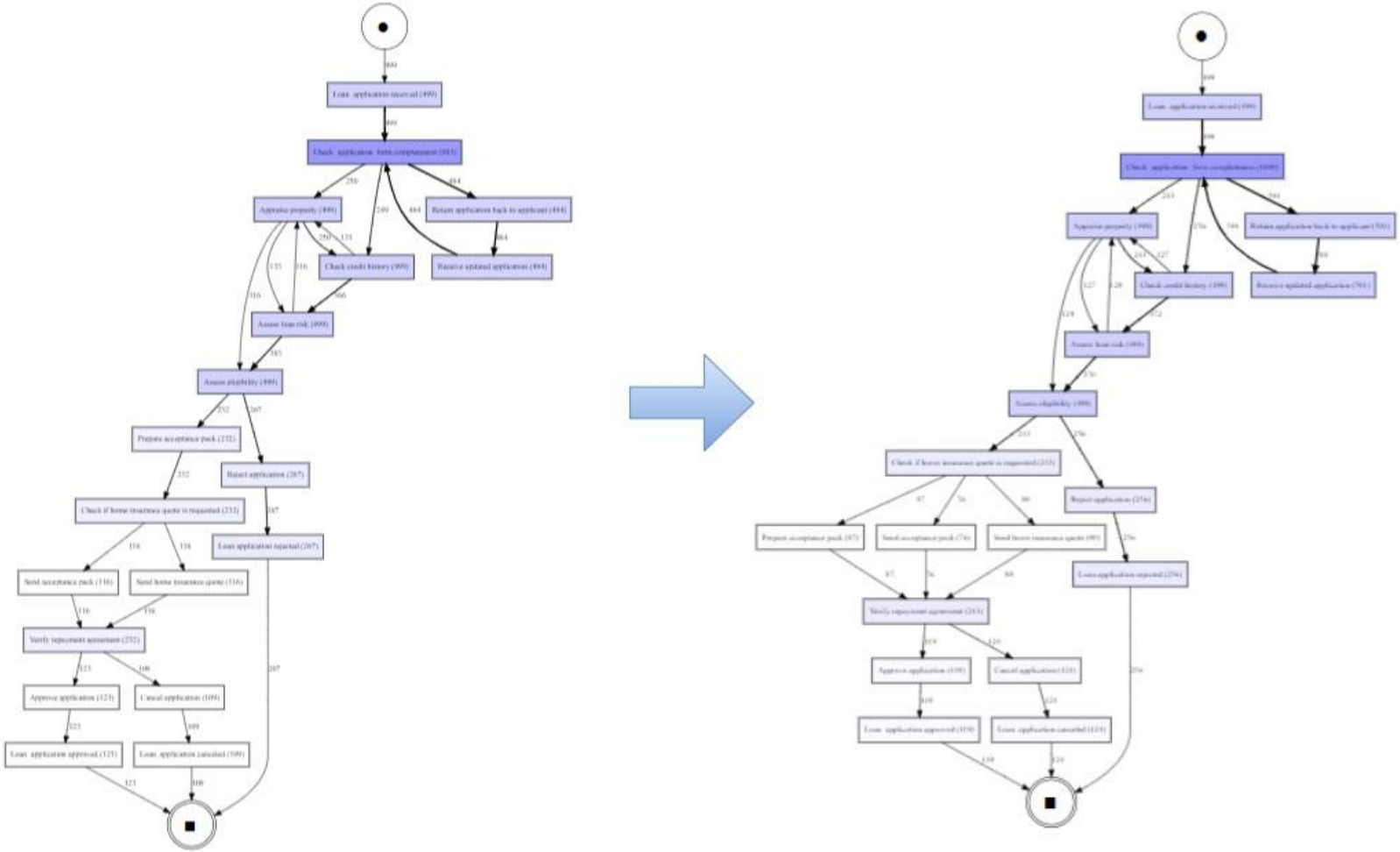
YUAN, J. et al. Detecting Sudden Drift for Single-Instance Log in Software Development Process. **Proceedings - 2020 Prognostics and Health Management Conference, PHM-Besancon 2020**, p. 354–359, 2020.

ZELLNER, L. et al. **Concept Drift Detection on Streaming Data with Dynamic Outlier Aggregation**. (Springer, Ed.)Process Mining Workshops - 1st International Workshop on Streaming Analytics for Process Mining (SA4PM'20). **Anais**...Padua, Italy: Springer, 2020. Disponível em: <https://icpmconference.org/2020/>

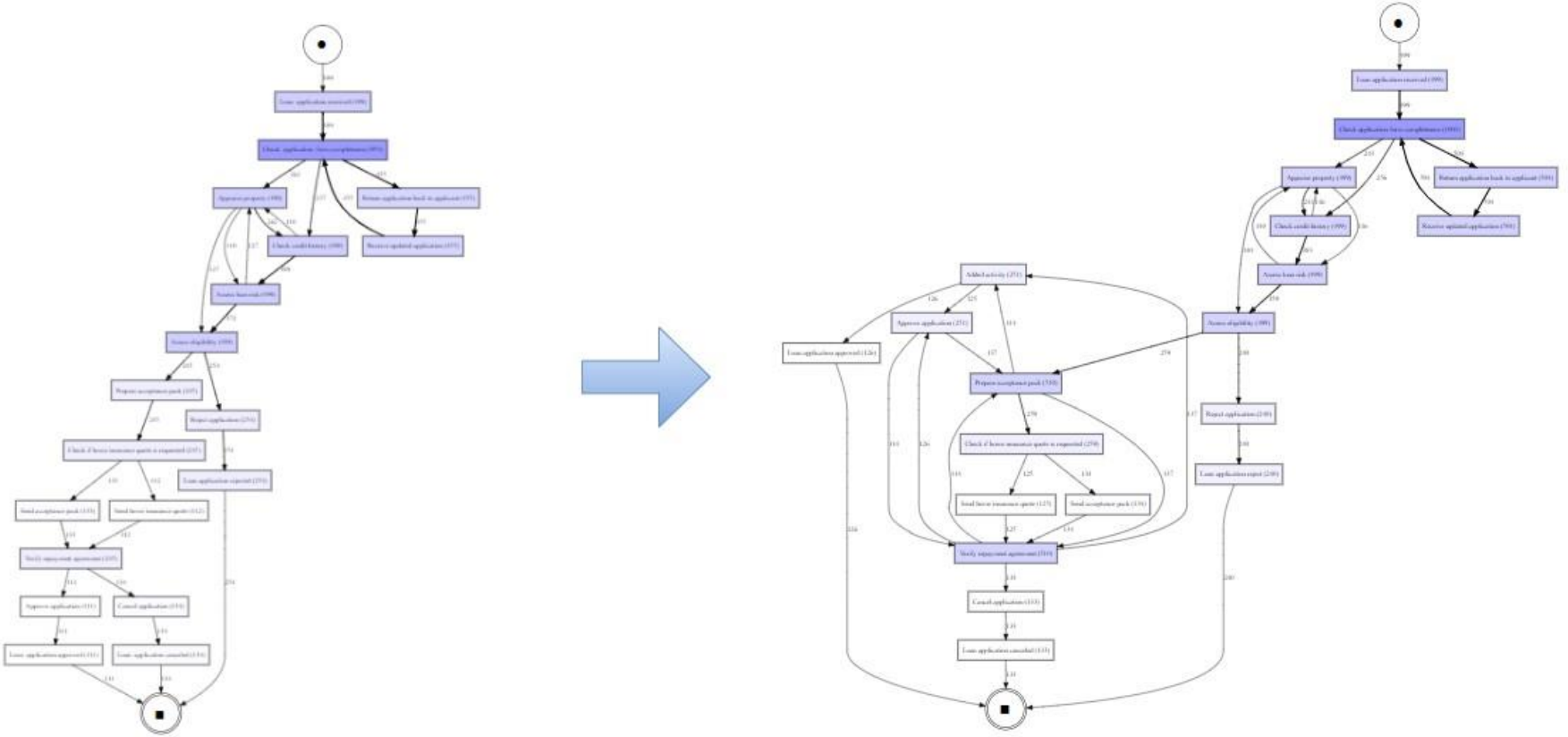
ZHENG, C.; WEN, L.; WANG, J. **Detecting process concept drifts from event logs**. (H.

Panetto et al., Eds.)Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). **Anais**...Cham: Springer International Publishing, 23 out. 2017. Disponível em: <https://link.springer.com/chapter/10.1007/978-3-319-69462-7_33>. Acesso em: 17 fev. 2021

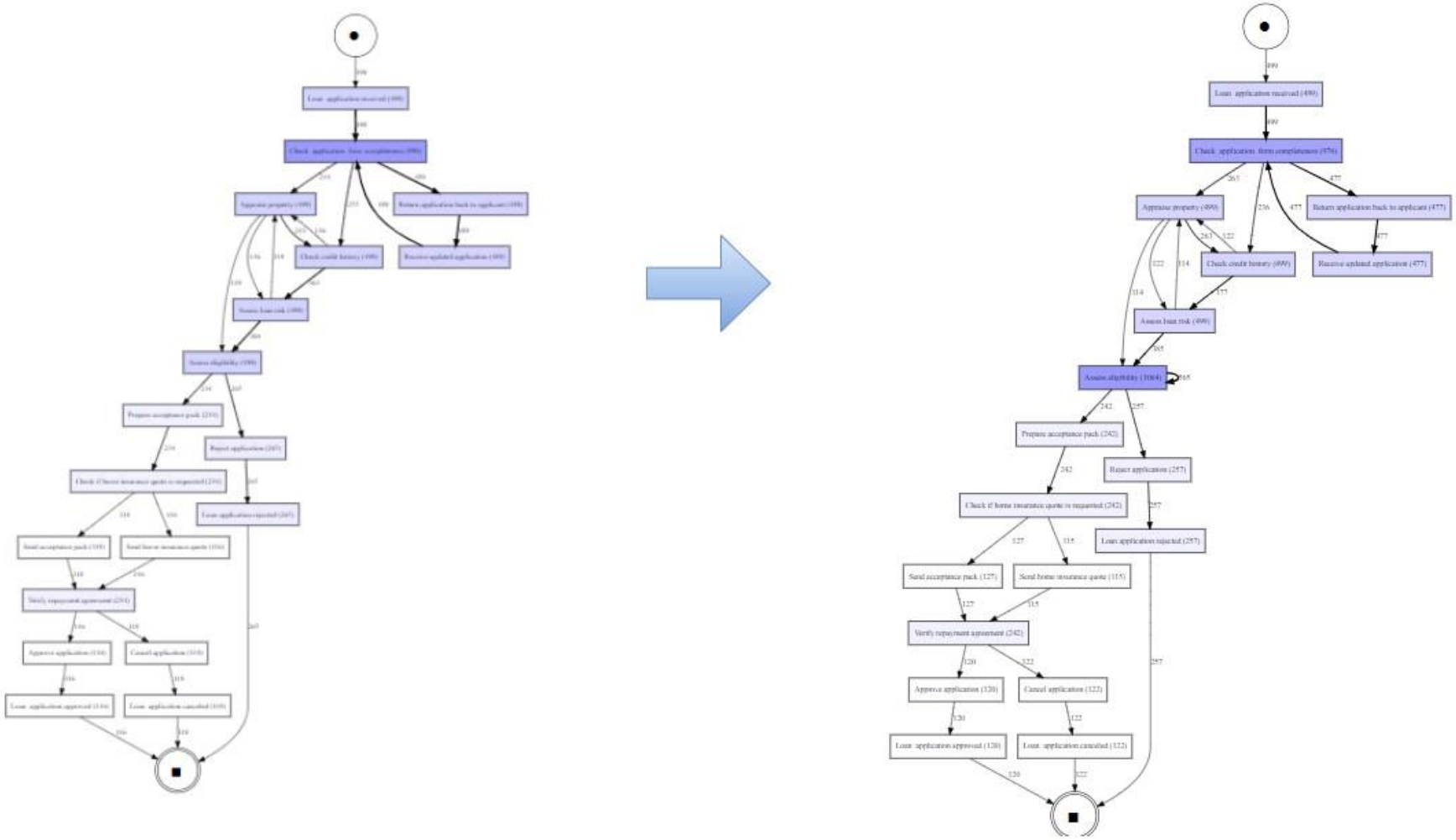
APÊNDICE A- Extração dos modelos da base gerada 2 (réplicas)



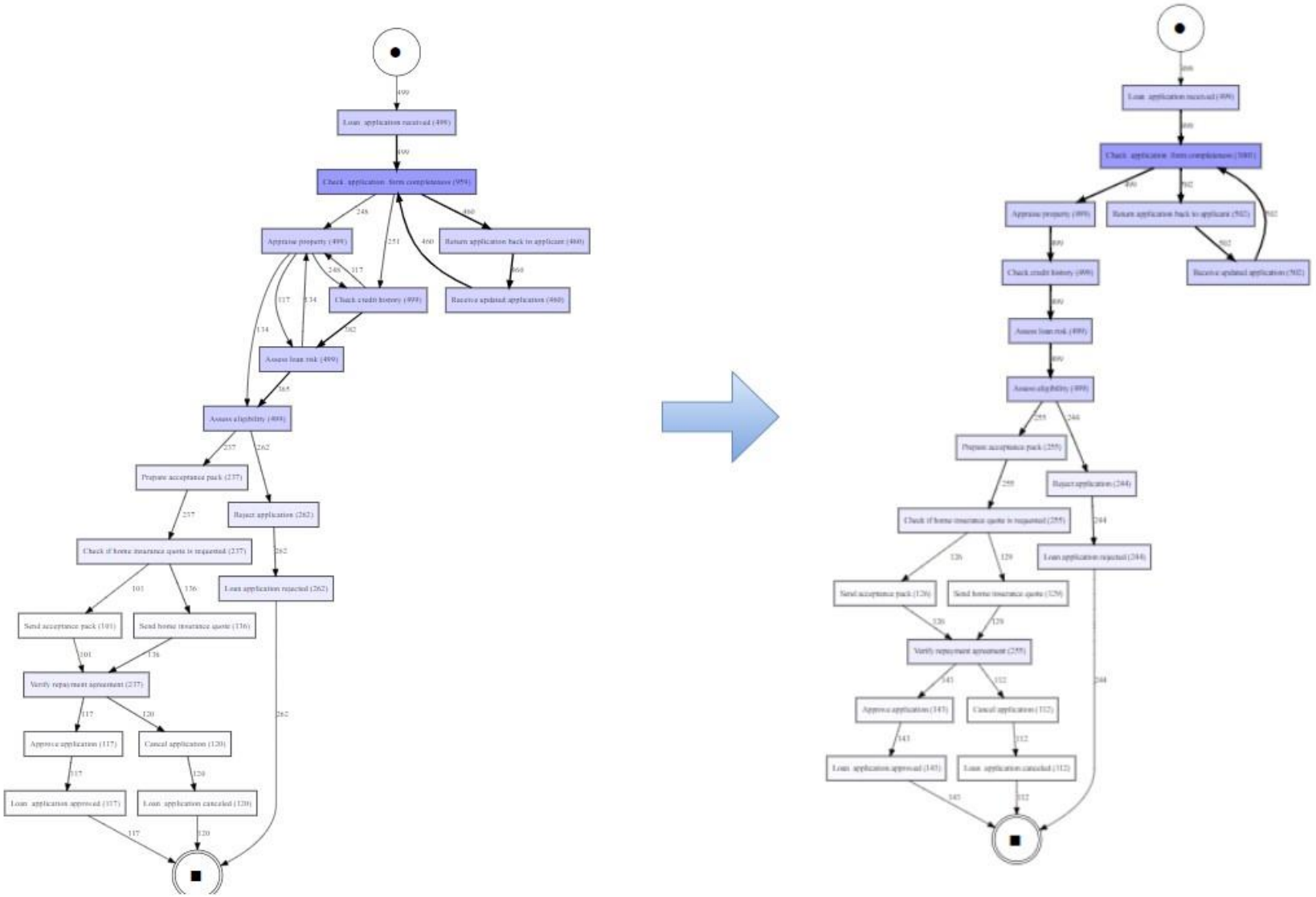
IRO5k_PDG_7



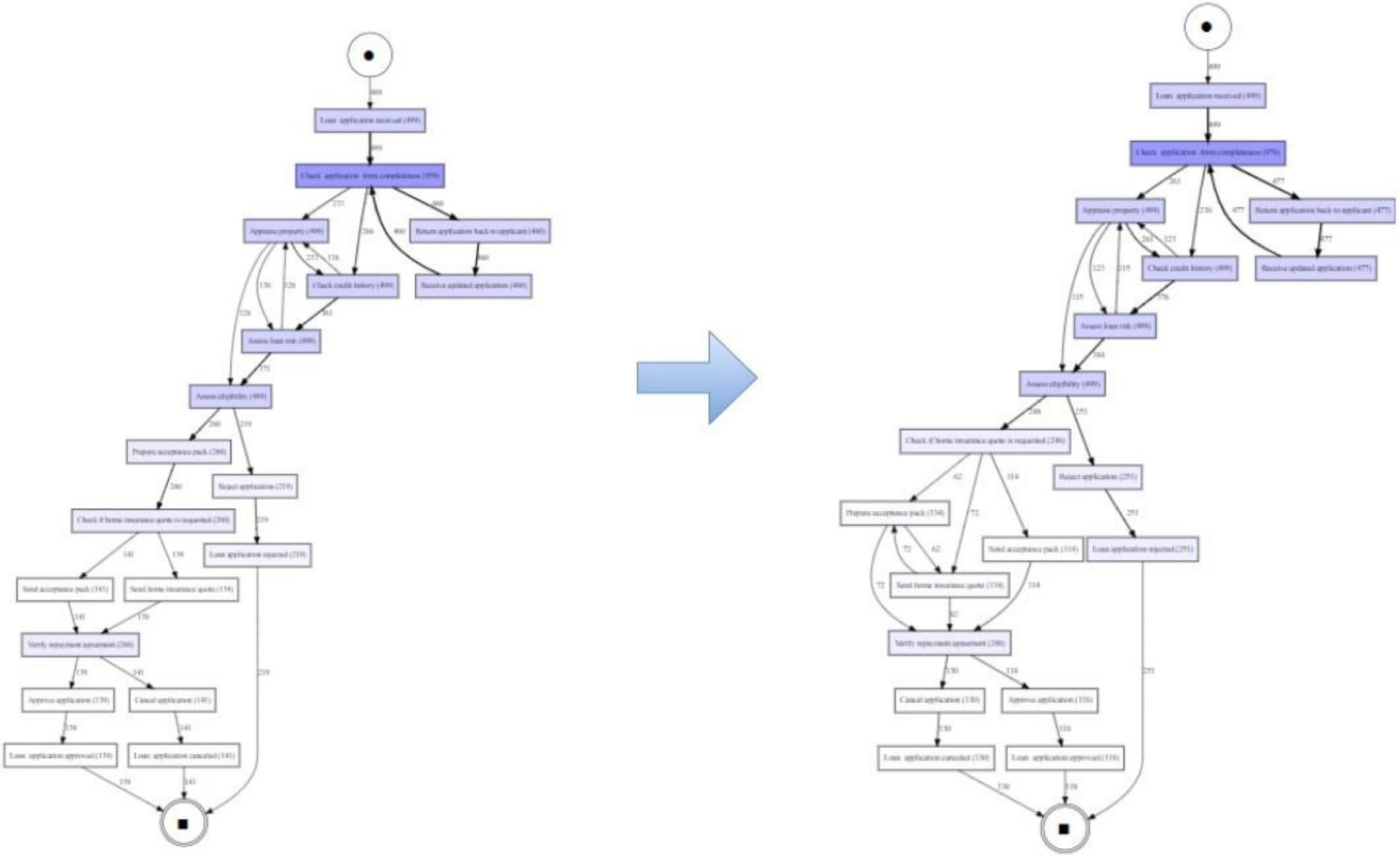
lp5k_PDG_4



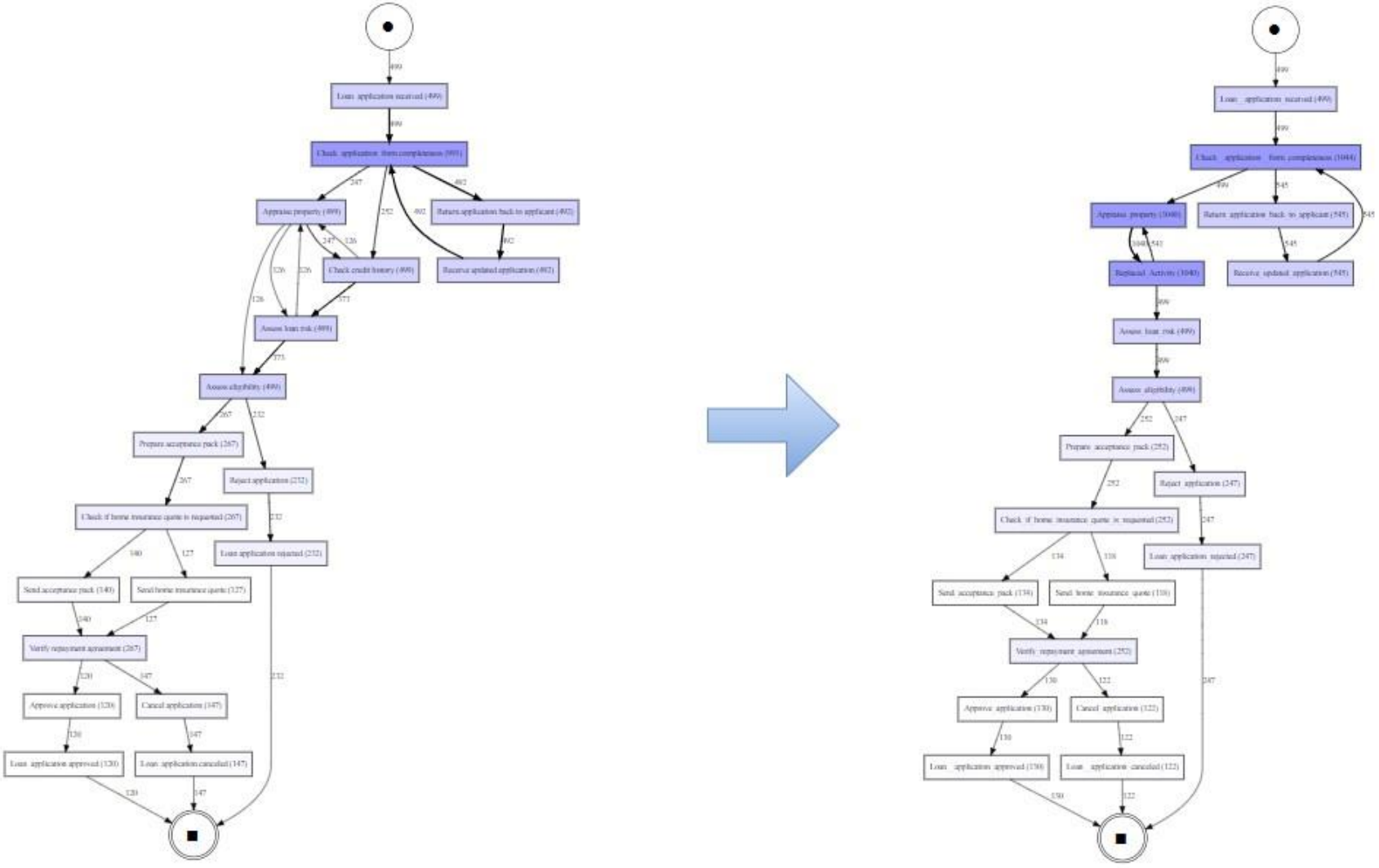
pl5k_PDG_4


Loan application received (499)
Check application form completeness (954)
Appraise property (499)
Return application back to applicant (460)
Check credit history (999)
Receive updated application (460)
Assess loan risk (499)
Assess eligibility (488)
Prepare acceptance pack (237)
Reject application (262)
Check if home insurance quote is requested (237)
Loan application rejected (262)
Send acceptance pack (101)
Send home insurance quote (136)
Verify repayment agreement (237)
Approve application (117)
Cancel application (120)
Loan application approved (117)
Loan application cancelled (120)
Loan application received (496)
Check application form completeness (1001)
Appraise property (499)
Return application back to applicant (502)
Check credit history (999)
Receive updated application (502)
Assess loan risk (488)
Assess eligibility (499)
Prepare acceptance pack (255)
Reject application (244)
Check if home insurance quote is requested (255)
Loan application rejected (244)
Send acceptance pack (126)
Send home insurance quote (129)
Verify repayment agreement (255)
Approve application (143)
Cancel application (112)
Loan application approved (143)
Loan application cancelled (112)

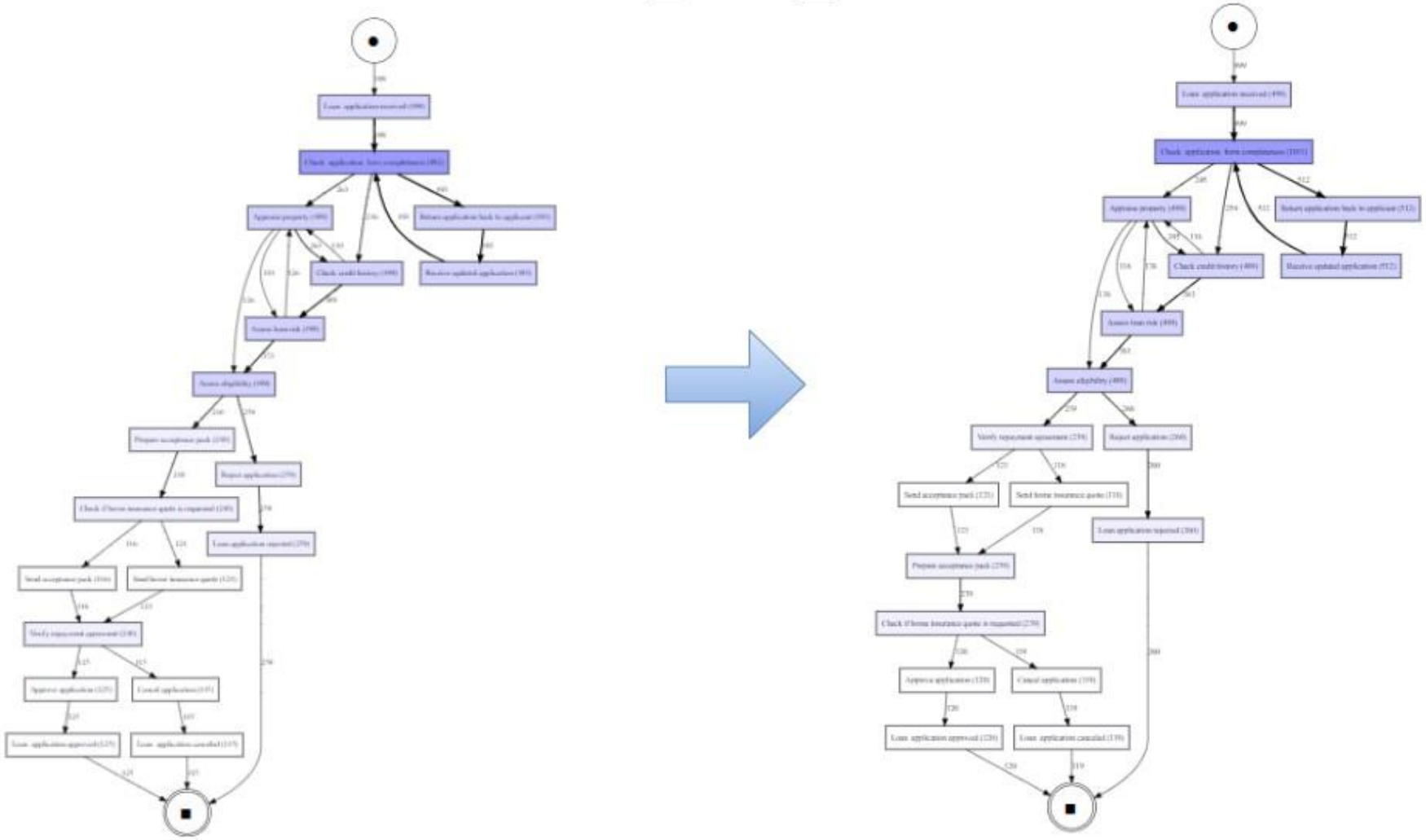
pm5k_PDG_3



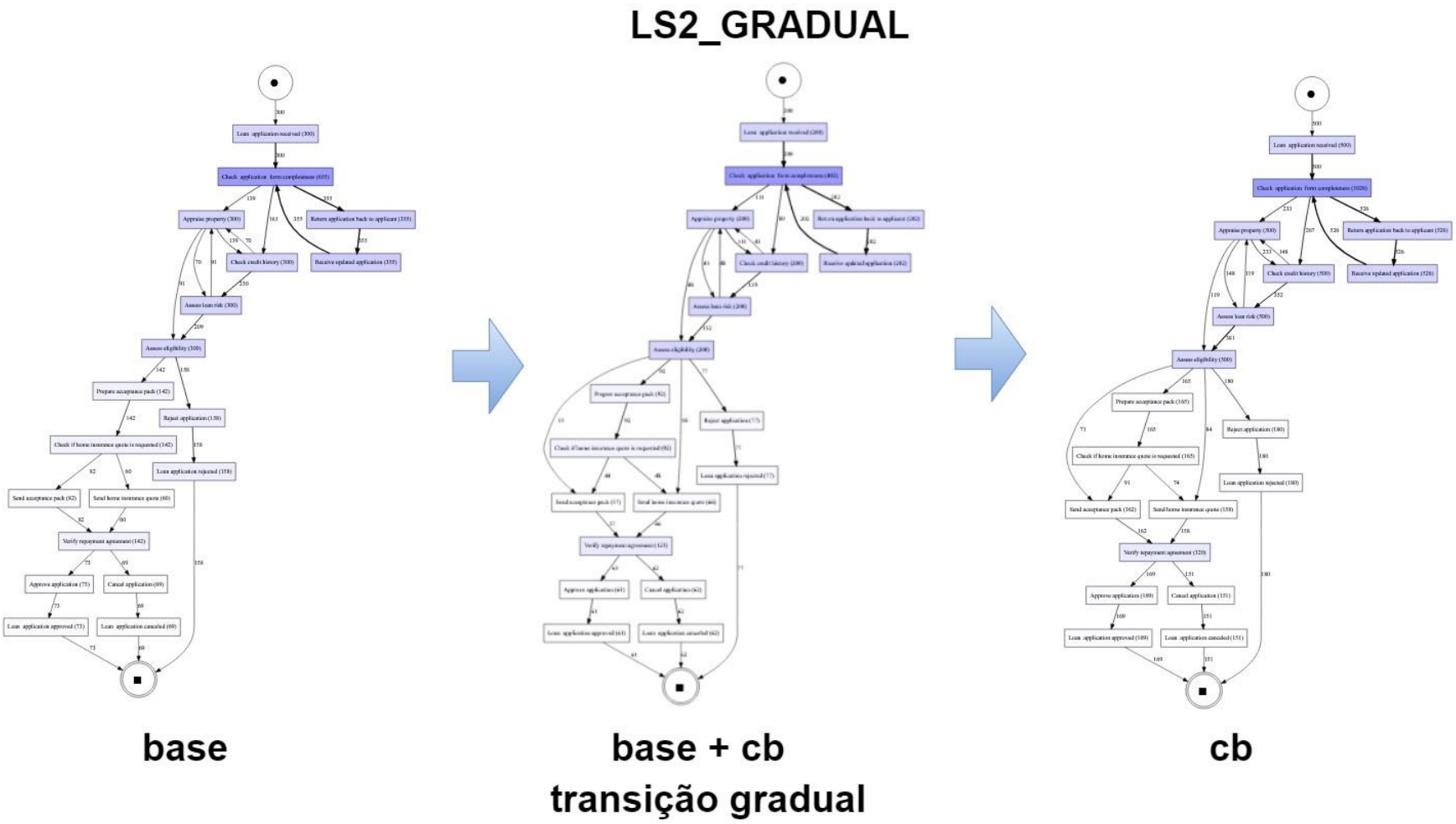
ROI5k_PDG_4


Loan application received (499)
Check application form completeness (499)
Appraise property (499)
Return application back to applicant (492)
Check credit history (499)
Receive updated application (492)
Assess loan risk (499)
Assess eligibility (499)
Prepare acceptance pack (267)
Reject application (232)
Check if home insurance quote is requested (267)
Loan application rejected (232)
Send acceptance pack (140)
Send home insurance quote (127)
Verify repayment agreement (267)
Approve application (120)
Cancel application (147)
Loan application approved (120)
Loan application cancelled (147)
Loan application received (499)
Check application form completeness (1044)
Appraise property (1040)
Return application back to applicant (545)
Replaced Activity (1040)
Receive updated application (545)
Assess loan risk (499)
Assess eligibility (499)
Prepare acceptance pack (252)
Reject application (247)
Check if home insurance quote is requested (252)
Loan application rejected (247)
Send acceptance pack (134)
Send home insurance quote (118)
Verify repayment agreement (252)
Approve application (130)
Cancel application (122)
Loan application approved (130)
Loan application cancelled (122)

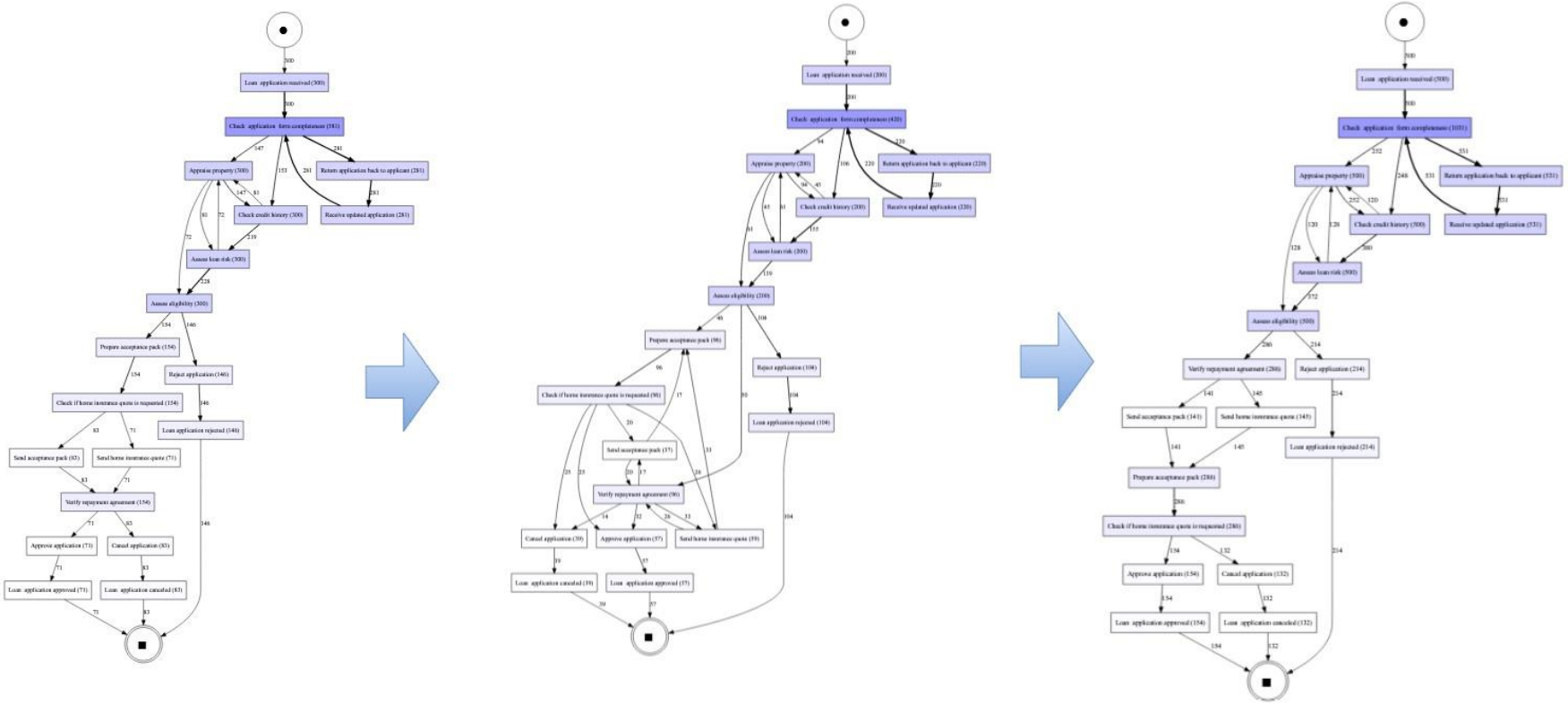
sw5k_PDG_4



APÊNDICE B- Modelos extraídos dos logs com transição gradual da Base Gerada 2



LS5_GRADUAL

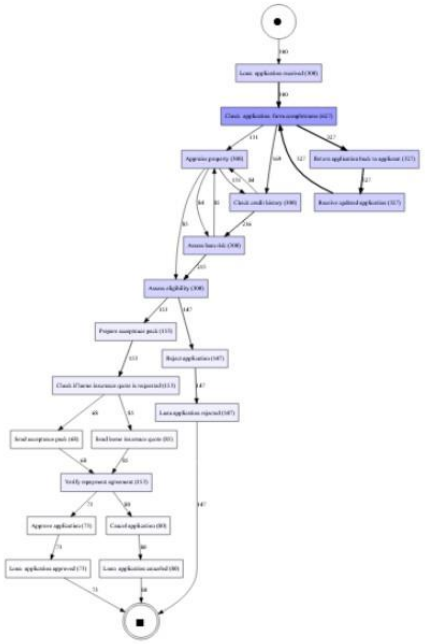


base

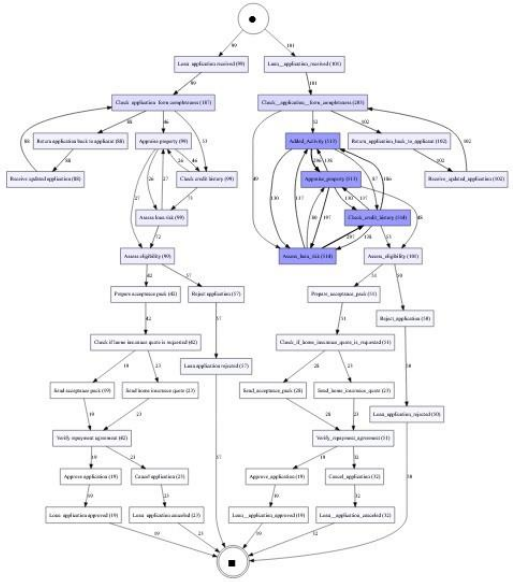
base + sw
transição gradual

sw

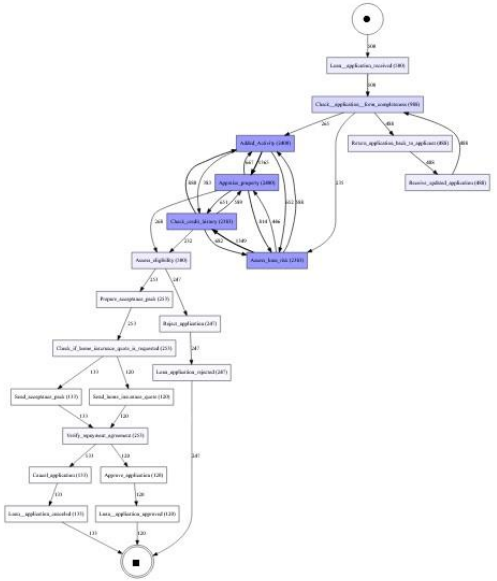
LS9_GRADUAL



base

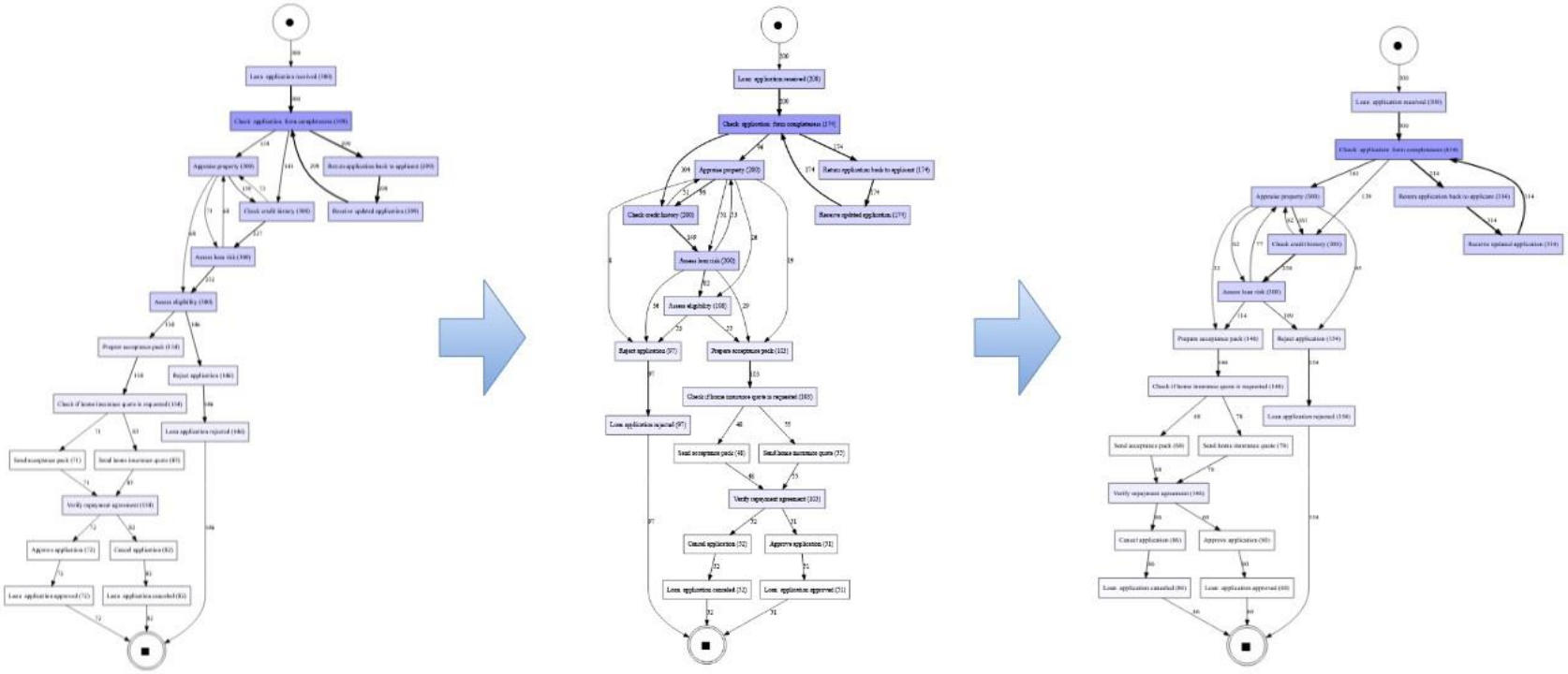


base + OIR
transição gradual



OIR

LS13_GRADUAL

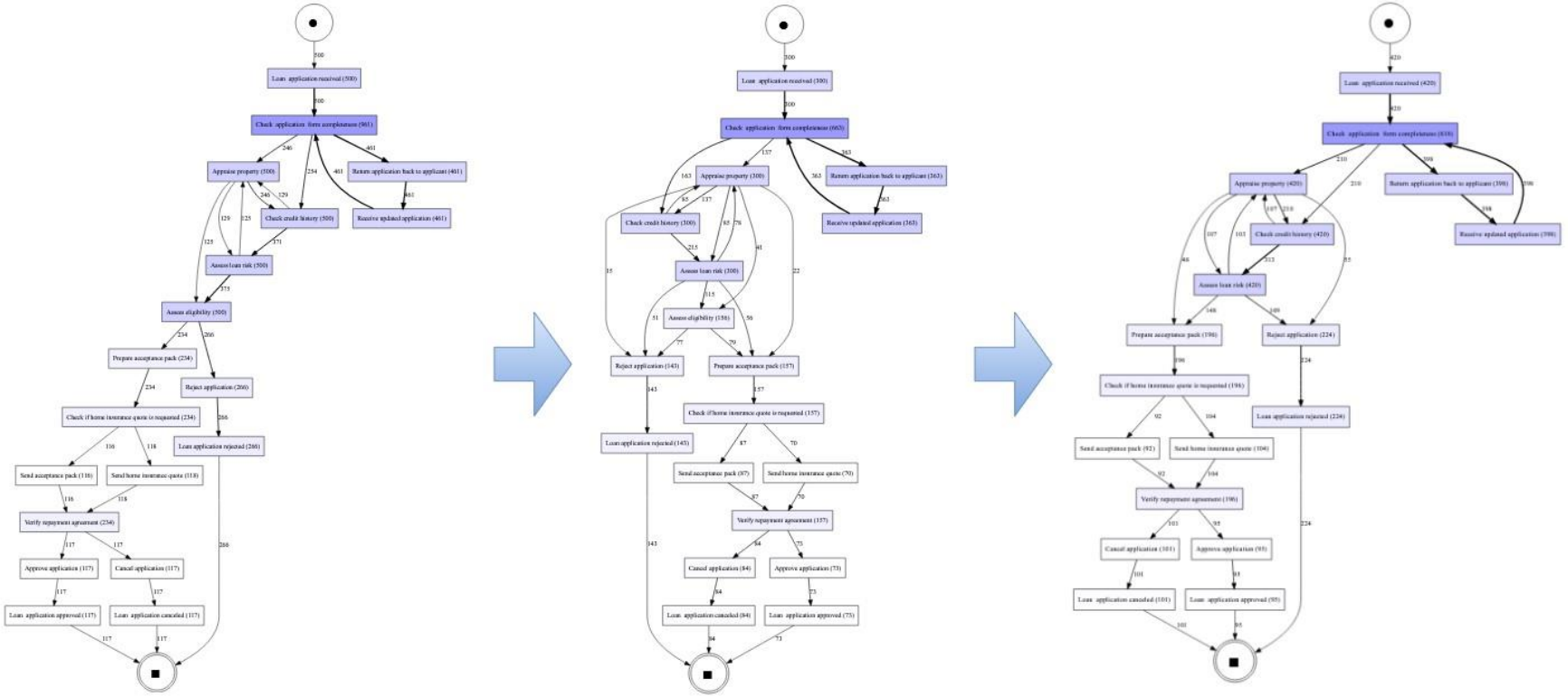


base

base + re
transição gradual

re

LS17_GRADUAL



base | **base + re transição gradual** | **re**

APÊNDICE C- *F-score* validação dos logs com transição abrupta da Base Gerada 2

Apromore - *ProDrift fixed*								
Event log	Window size							
	32	50	75	100	125	150	162	200
LS1_ABRUPTO	0	1	1	1	1	1	1	1
LS10_INCREMENTAL	0,5	0,5	0,5	0,8	0,8	0,8	0,8	0,8
LS11_RECORRENTE	0,8	1	1	1	1	1	1	1
LS12_ABRUPTO	0,857142857	1	1	1	1	1	1	1
LS14_INCREMENTAL	0,4	0,666666667	0,857142857	0,857142857	0,857142857	0,8571429	0,857142857	0,857142857
LS15_RECORRENTE	1	1	0,923076923	1	1	1	1	1
LS16_INCREMENTAL	0,153846154	0,24	0,272727273	0,4	0,4	0,4615385	0,4	0,666666667
LS3_INCREMENTAL	0,666666667	0,666666667	0,666666667	0,666666667	0,666666667	1	0,666666667	0,666666667
LS4_RECORRENTE	0	0,666666667	0,666666667	0,666666667	1	1	1	1
LS6_INCREMENTAL	0,666666667	0,666666667	1	1	1	1	1	1
LS7_RECORRENTE	0	1	1	1	1	1	1	1
LS8_ABRUPTO	1	1	1	1	1	1	1	1
MergedLog_1	0,612903226	0,64516129	0,631578947	0,76	0,76	0,7916667	0,844444444	0,863636364
MergedLog_2	0,545454545	0,857142857	0,857142857	0,857142857	0,933333333	1	0,933333333	0,769230769
MergedLog_3	0,666666667	0,909090909	0,909090909	0,909090909	0,909090909	0,9090909	0,909090909	0,909090909
Apromore - ProDrift fixed - dataset 4	0,524623119	0,787870782	0,81893954	0,861113997	0,888415584	0,9212959	0,894045214	0,902162282

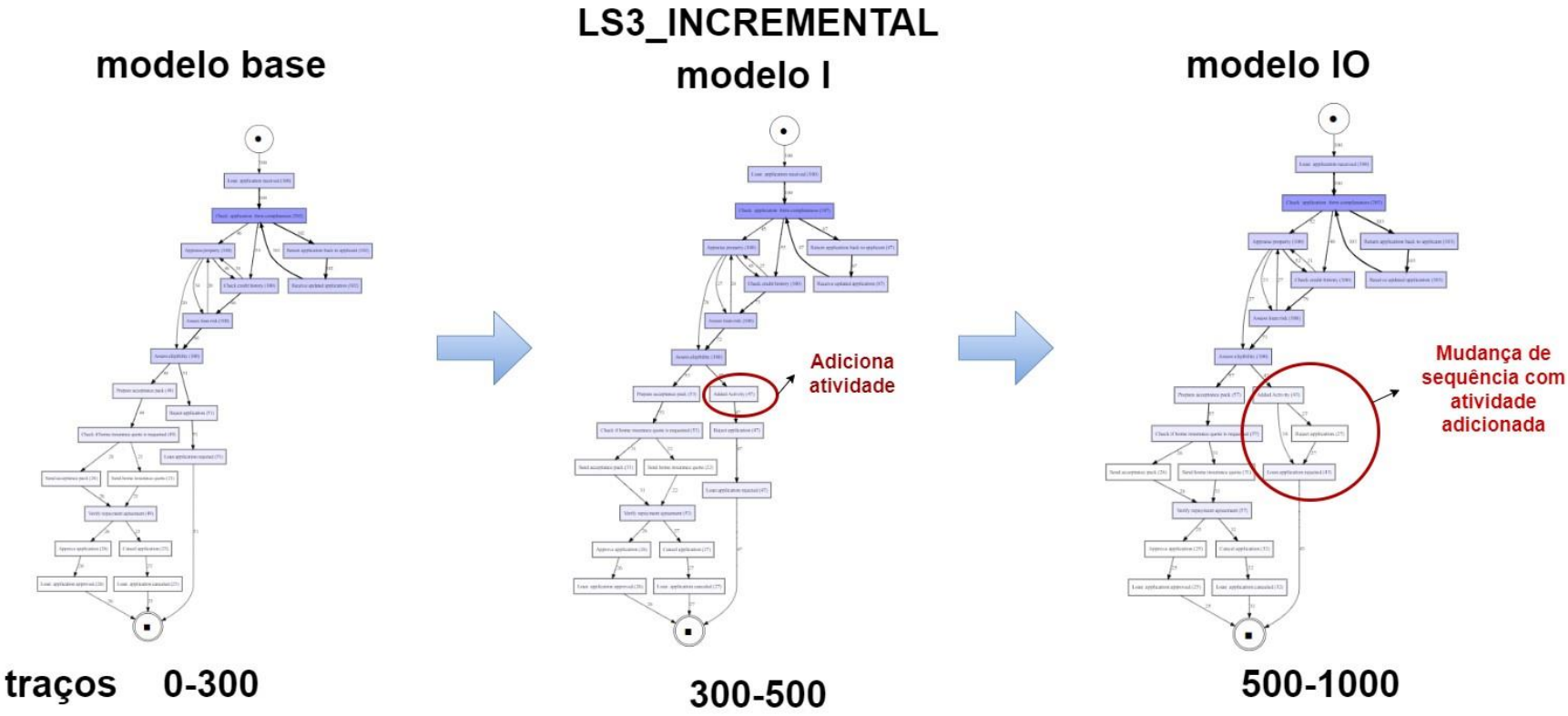
VDD

Event log	***Window size***							
	32	**50**	**75**	**100**	**125**	**150**	**162**	**200**
LS1_ABRUPTO	0,4	0	0,4	0,4	0,8	0,5	0,5	0,5
LS10_INCREMENTAL	0	0,33333333	0,33333333	0,33333333	0,66666667	0,33333333	0,33333333	0,66666667
LS11_RECORRENTE	0,28571429	0,57142857	0,28571429	0,28571429	0,85714286	0,28571429	0,28571429	0,57142857
LS12_ABRUPTO	0	0,22222222	0,22222222	0,22222222	0,22222222	0,25	0,5	0,5
LS14_INCREMENTAL	0,5	0,75	0,25	0,75	0,25	0	0,5	0,75
LS15_RECORRENTE	0,19047619	0,22222222	0,19047619	0,19047619	0,11111111	0	0,22222222	0,22222222
LS16_INCREMENTAL	0	0,75	0	0,75	0	0,5	0	0,75
LS3_INCREMENTAL	0,25	0,33333333	0,66666667	0,4	0,5	0,5	0,5	0,5
LS4_RECORRENTE	0,4	0,4	0,66666667	0,33333333	0,5	0	0,5	0,5
LS6_INCREMENTAL	0	0,8	0	0,4	0,4	0,33333333	0	0,4
LS7_RECORRENTE	0	0,8	0	0,4	0,4	0,4	0	0,4
LS8_ABRUPTO	0,28571429	0,57142857	0,28571429	0,28571429	0,85714286	0,28571429	0,28571429	0,57142857
MergedLog_1	0	0	0	0	0	0,07142857	0,21428571	0,22222222
MergedLog_2	0,15384615	0,46153846	0,30769231	0,46153846	0	0,18181818	0,18181818	0,36363636
MergedLog_3	0,16666667	0,66666667	0,33333333	0,2	0	0,4	0,2	0,4
VDD **- base gerada 2**	**0,17549451**	**0,45881156**	**0,26278795**	**0,36082214**	**0,37095238**	**0,2694228**	**0,2815392**	**0,48784031**

IPDD *framework*

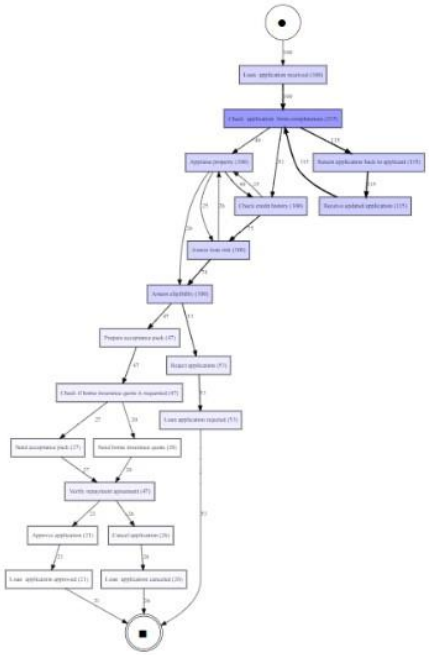
Event log	***Window size***							
	32	**50**	**75**	**100**	**125**	**150**	**162**	**200**
LS1_ABRUPTO	0,4	1	0,8	1	0,5	0,8	0,4	0,8
LS10_INCREMENTAL	0,117647	0,75	1	1	0,571429	1	0,285714	0,285714
LS11_RECORRENTE	0,666667	1	1	1	0,75	1	0,666667	0,666667
LS12_ABRUPTO	0,666667	0,888889	0,727273	0,888889	0,8	0,727273	0,666667	0,727273
LS14_INCREMENTAL	0,206897	0,615385	0,6	1	1	0,6	0,363636	1
LS15_RECORRENTE	0,684211	0,962963	0,742857	1	1	0,742857	0,666667	1
LS16_INCREMENTAL	0,057143	0,296296	0,375	0,666667	0,8	0,6	0,4	0,75
LS3_INCREMENTAL	0,4	1	0,5	1	0,8	0,5	0,4	0,5
LS4_RECORRENTE	0,4	1	1	1	0,5	1	0,5	0,5
LS6_INCREMENTAL	0,4	1	0,5	1	0,8	0,5	0,4	1
LS7_RECORRENTE	0,666667	1	0,8	1	0,8	0,8	0,666667	1
LS8_ABRUPTO	0,666667	1	1	1	0,75	1	0,666667	0,666667
MergedLog_1	0,321429	0,634921	0,666667	0,851064	0,851064	0,740741	0,612903	0,826087
MergedLog_2	0,545455	1	0,7	1	0,705882	0,7	0,526316	0,705882
MergedLog_3	0,588235	0,909091	0,666667	0,909091	0,714286	0,666667	0,625	0,714286
IPDD - base gerada 2	**0,452512**	**0,870503**	**0,738564**	**0,954381**	**0,756177**	**0,758502**	**0,523127**	**0,742838**

APÊNDICE D- Evolução dos modelos para a validação da evolução dos modelos no *drifts* incrementais e recorrentes



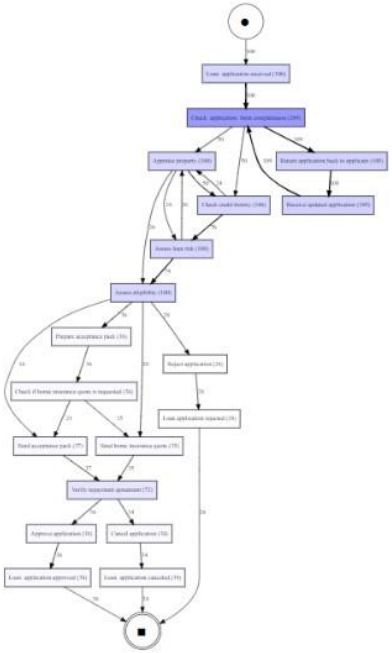
LS4_RECORRENTE

modelo base



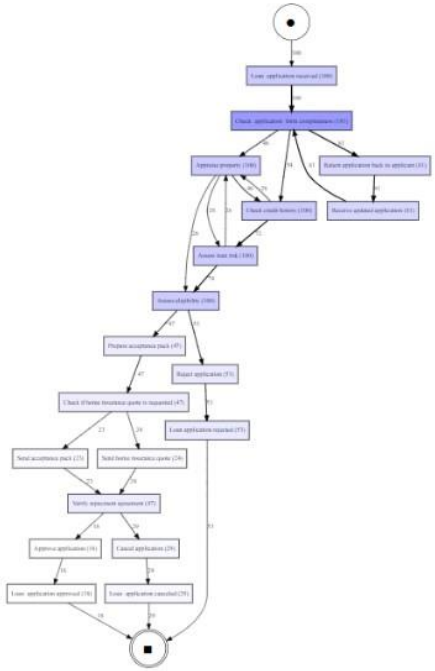
traços 0-300

modelo cb



300-500

modelo base



500-1000