Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Método de Mineração de Processos para Auxílio à Tomada de Decisão: um Estudo de Caso no Controle de Férias

Pablo Ignacio Gandulfo

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2016

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Célia Ghedini Ralha

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof.ª Dr.ª Maria Emília Machado Telles Walter — CIC/UnB
Prof.ª Dr.ª Fernanda Araujo Baião Amorim — UNIRIO

CIP — Catalogação Internacional na Publicação

Gandulfo, Pablo Ignacio.

Método de Mineração de Processos para Auxílio à Tomada de Decisão: um Estudo de Caso no Controle de Férias / Pablo Ignacio Gandulfo. Brasília : UnB, 2016.
113 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2016.

1. Órgão Público, 2. Gestão de Pessoas, 3. Apoio à Tomada de Decisão

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Pablo Ignácio Gandulfo

Método de Mineração de Processos para Auxílio à Tomada de Decisão: Um Estudo de Caso no Controle de Férias

Dissertação aprovada como requisito parcial para obtenção do grau de **Mestre** no Curso de Pós-graduação em Informática da Universidade de Brasília, pela Comissão formada pelos professores:

Orientador: Prof.ª Dr.ª Célia Ghedini Ralha, CIC/UNB

Prof.ª Dr.ª Maria Emília Machado Telles Walter, CIC/UNB

Prof.ª Dr.ª Fernanda Araújo Baião Amorim, UNIRIO

Dr.ª Vanessa Tavares Nunes, PPGI/CIC/UNB

Vista e permitida a impressão.
Brasília, 21 de Julho de 2016.

Prof.ª Dr.ª Célia Ghedini Ralha
Coordenadora do Programa de Pós-Graduação em Informática
Departamento de Ciência da Computação
Universidade de Brasília.

Universidade de Brasília

Dedicatória

A minha esposa Daniela, pelo apoio e companheirismo oferecido ao longo desses anos. Aos meus filhos Roberto, Mariana e Mateus, por existirem e me iluminarem com a sua alegria e carisma.

A minha mãe Ana Maria e meu pai Roberto, que me deram educação e guiaram desde o dia em que nasci.

Agradecimentos

Agradeço

Aos meus pais, Roberto e Ana Maria, pelo apoio incondicional e suporte familiar, fundamental para que eu pudesse dispor de tempo e foco para realizar este trabalho. Aos meus familiares (irmãs, cunhada e marido, tios, tias, primos, primas), que me deram bons conselhos em vários momentos importantes da minha vida.

A minha amiga, professora e orientadora Dra. Célia Ghedini Ralha, que acreditou em mim, incentivou e direcionou a realização de toda esta pesquisa, além de fornecer valioso conhecimento ao longo de todo o curso. Meu empenho e esforço foram investidos em agradecer, de alguma forma, a sua generosidade ao me acolher como seu orientando.

A Dra. Vanessa Tavares Nunes, por ter prestado um apoio de grande importância na realização deste trabalho. Seu conhecimento, orientação e sugestões construtivas tornaram a caminhada mais suave e contínua.

Aos professores da UnB, que foram tão dedicados na função de educar e transmitir da melhor forma possível o conteúdo necessário. A UnB, pela oportunidade em realizar este curso de mestrado em Ciência da Computação.

Aos meus amigos Daniela, Mauricio, Deborah, Guilherme, Karina e Christian, por terem me apoiado em vários momentos ao longo dessa jornada, principalmente nos mais difíceis. Sem eles eu não teria conseguido.

Aos colegas de turma e do grupo de pesquisa InfoKnow - Computer Systems for Information and Knowledge Treatment, que mantiveram um clima harmonioso e um grupo unido até o último semestre deste curso, uns ajudando os outros sempre que necessário.

Por fim, agradeço a todos que acreditaram em mim, e a todos que porventura não foram aqui mencionados, mas que de alguma forma me ajudaram nessa gratificante jornada.

Resumo

A prática de mineração de processos tem sido impulsionada pela necessidade das organizações conhecerem cada vez mais os seus processos de negócio. No entanto, ainda existem alguns desafios e obstáculos a serem superados. Pode-se citar a escassez de modelos e métodos que efetivamente ajudem as organizações a reduzir custos e otimizar o uso de recursos nos fluxos de trabalho organizacionais. Neste cenário, o objetivo geral desta pesquisa é descrever um modelo de mineração de processos para o apoio à tomada de decisões nas organizações, aplicando-o no âmbito do processo de gestão de pessoas nas organizações públicas. O método de mineração desenvolvido foi definido através da junção dos métodos de Bozkaya et al. (2009) e van der Ham (2015), e ilustrado através de uma descrição detalhada dos passos envolvidos. O método foi aplicado em um estudo de caso real numa organização judicial. Foi descoberto um retrabalho de 4,3 vezes na tarefa "Alterar Homologação de Férias". Além disso, o papel "Analista de RH" está consideravelmente sobrecarregado, pois, atua em 47,05% das atividades realizadas, apesar de representar apenas 3,45% dos recursos envolvidos. Esses indicadores de desperdício permitem uma tomada de decisão do gestor para melhoria no processo de controle de férias da organização estudada.

Palavras-chave: Órgão Público, Gestão de Pessoas, Apoio à Tomada de Decisão

Abstract

The increase of process mining's practice has been driven by the need from organizations to better understand their processes. On the other hand, there are still some challenges and obstacles to be overcome. One could mention the lack of models and methods that effectively help government organizations to reduce costs and optimize the use of resources on their personnel management's workflows. In that respect, the objective of this research is to describe a model for process mining to support decision-making in organizations, applying it in the personnel management's processes at government organizations. The mining method developed was defined by a combination of the methods of Bozkaya et al. (2009) and van der Ham (2015), with a detailed description of the several steps involved. The method was applied in a real case study at a government organization from the judicial area. A 4.3 times rework was discovered on "Change Vacation approval" task. Besides that, the role "HR Analist" is considerably overloaded because it acts in 47.05% of all activities in spite of representing 3.45% of all resources involved. This waste indicators allow for the decision-making of the manager to improve the vacation control process of the organization analyzed.

Keywords: Public Agency, Personnel Management, Support Decision-Making

Abreviaturas

BI - *Business Intelligence.*

BPIC - *Business Process Intelligence Challenge.*

BPM - *Business Process Modeling.*

BPMN - *Business Process Model and Notation.*

CLT - Consolidação das Leis do Trabalho.

CSV - *Comma-separated values.*

e-Commerce - *Electronic Commerce* (Comércio Eletrônico).

ERP - *Enterprise Resource Planning.*

ID - *Identification.*

IEEE - *Institute of Electrical and Electronics Engineers.*

JDBC - *Java Database Connectivity.*

MP - Mineração de Processos.

MXML - *Mining eXtensible Markup Language.*

ProM - *Process Mining framework.*

RH - Recursos Humanos.

SI - Sistemas de Informação.

SGBD - Sistema de Gerenciamento de Banco de Dados.

SQL - *Structured Query Language.*

UnB - Universidade de Brasília.

XES - *eXtensible Event Stream.*

XML - *eXtensible Markup Language.*

XOR - *eXclusive OR* (Disjunção Exclusiva).

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Existe um considerável desperdício de recursos na maioria das organizações, decorrente em grande parte de falhas no planejamento, controle e acompanhamento dos trabalhos realizados. Por outro lado, o consumidor está cada vez mais exigente, não só pelo fácil acesso à informação e consciência de seus direitos e deveres, mas também pela grande oferta existente e a concorrência cada vez mais assídua entre os fornecedores. Aspectos como rapidez e praticidade no atendimento, economia de tempo e dinheiro, além de qualidade nos serviços prestados, têm sido cada vez mais valorizados pelo público consumidor.

No cerne dessa questão está a importância que os Sistemas de Informação (SI) desempenham na gestão e operacionalização dos processos organizacionais (Laudon and Laudon, 2015). Desde a concepção até a implantação e acompanhamento de seus processos, as organizações dependem cada vez mais de sistemas inteligentes, integrados e que se adequam às necessidades do usuário, permitindo gerir os negócios em tempo real, munir os usuários de informações relevantes para a apropriada tomada de decisão, além de maximizar os resultados e ganhos. Um exemplo desses sistemas são os conhecidos sistemas de Planejamento de Recurso Corporativo (Enterprise Resource Planning - ERP), que contemplam praticamente todos os fluxos de trabalho, atividades e recursos, sejam eles humanos ou materiais, dentro das organizações. Outro exemplo são os de Comércio Eletrônico (*e-Commerce*), que lidam com os diversos aspectos envolvidos nas transações, entre empresas e/ou indivíduos, de forma eletrônica. Tanto nos casos de grande dinamismo e complexidade, como um tratamento médico de um paciente, quanto de relativa estabilidade e previsibilidade, como uma produção em linha de montagem, os SI não só permitem uma adequada gestão dos processos como entendem cada vez mais o conceito de processo em si, utilizando-o para modelar fluxos, relacionar pessoas com recursos, estados e possíveis ações que podem ser executadas.

Vários SI geram registros de histórico de execução (*logs*) contendo informações sobre o que acontece nas várias instâncias de cada processo. Esses registros representam acontecimentos atômicos, denominados eventos, e podem referenciar o técnico ou grupo de técnicos envolvidos, uma descrição da atividade realizada, a data e hora em que ocorreu, entre outros. Esses eventos, apesar de não agregarem muito valor quando vistos de forma individual para o objetivo em questão, quando analisados em conjunto podem ser de grande utilidade no entendimento ou até mesmo na melhoria e otimização dos processos organizacionais.

Uma das necessidades relacionadas com a análise desses eventos é a de levantar uma

visão mais abrangente e de alto nível do processo completo. Como muitas organizações estão subdivididas em departamentos e projetos, cada um operando com algum nível de autonomia e complexidade, praticamente nenhum *stakeholder*, se é que existe algum, possui uma visão ampla e clara de como o processo funciona como um todo. Portanto, podem ser utilizados métodos de mineração nesses eventos, através da análise e exploração, que permitam fazer engenharia reversa e montar uma visão de alto nível e completa do processo organizacional.

Outra necessidade comum é a de validar se as atividades estão sendo realizadas conforme um processo previamente planejado e definido. Pode ser que um ou mais dos fluxos de trabalho estabelecidos não estejam sendo seguidos conforme esperado, tanto por uma questão acidental, como uma falha no modelo ou uma configuração errada de sistema, quanto até mesmo intencional, motivada por fraude ou outra finalidade.

Todas essas aplicações, somadas, tornam a atividade de mineração de processos (MP) de grande importância para uma gestão de excelência nos processos de uma organização. A MP é uma prática que permite aprofundar a análise de processos através de técnicas e ferramentas específicas para a descoberta de processos, controles, dados e estruturas sociais e organizacionais (Saylam and Sahingoz, 2013). A idéia básica da análise de processos é estudá-los através da mineração dos *logs* de eventos gerados pelos SI envolvidos. Uma das grandes vantagens da MP é a objetividade, já que os dados utilizados para análise são reais e precisos. Além disso, ela permite avaliar outras questões, como conformidade no processo, encontrar exceções e identificar possíveis causas relacionadas.

Se por um lado a MP possui grande importância, existem alguns desafios intrínsecos à essa disciplina que, no geral, não são simples de enfrentar. A própria técnica de mineração de dados por si só já apresenta alto nível de complexidade. E quando aplicada no âmbito de processos organizacionais, novos conceitos, variáveis e obstáculos surgem, dificultando ainda mais a busca e extração de informações relevantes para tomada de decisão.

Considerando a existência de uma grande variedade de algoritmos e técnicas de mineração, é esperado que o especialista em mineração deseje adotar algum método que o auxilie e oriente nas diversas escolhas que deverá fazer para realizar os trabalhos adequadamente, independente do caso de uso em questão.

No entanto, é possível notar uma escassez de métodos que sejam aplicáveis a diferentes contextos de negócio, principalmente considerando que cada área de aplicação apresenta características peculiares de ambiente e logística de trabalho, além de que cada organização opera sob regras que podem ser consideradas únicas e singulares ao compará-la com as demais.

1.1 Motivação

O ramo da Ciência da Computação que trata de MP, iniciado aproximadamente em 1999 (van der Aalst, 2013b), conquistou importantes progressos ao longo dos últimos anos. É possível notar esse avanço através da grande quantidade de algoritmos desenvolvidos Yue et al. (2011), e do potencial das ferramentas disponíveis (Buijs, 2010; Szimanski, 2013). Por exemplo, o ProM, uma ferramenta de referência na área, é baseado num *framework* de *plugins* de algoritmos de mineração, com mais de 142 *plugins* na versão 4.0, 286 *plugins* na versão 5.2 e mais de 100 *plugins* na versão 6.4 (van der Aalst, 2016; van der Aalst et al., 2007).

Essa riqueza de algoritmos e ferramentas é de extrema importância para a análise de processos, controles, dados e estruturas organizacionais. No entanto, ainda existem várias questões em aberto nessa área que demandam pesquisa e estudo para tornar o trabalho de MP cada vez mais normatizado e sistemático.

Um dos desafios envolvidos é a escolha do modelo mais adequado de mineração para o caso específico sendo analisado. Bozkaya et al. (2009) aponta essa questão ao colocar que, apesar de existirem várias pesquisas sobre MP, um grande problema é que elas são feitas caso a caso, baseadas em percepções e conhecimento que o especialista em mineração possui sobre o caso de estudo. Além disso, os autores Bozkaya et al. (2009) deixam claro que não existe um método que lide com novos casos de estudo, ou seja, que é difícil tornar o método repetitível.

A escassez de um método mais abrangente torna o processo de aprendizagem complexo e trabalhoso, já que envolve o estudo de uma grande quantidade de métodos específicos, cada um aplicável para um caso ou um conjunto de casos particulares. Além disso, pode ser que ainda não exista um método já desenvolvido para o contexto específico sendo analisado, dificultando a realização da análise.

Além da escolha do método, outro ponto crucial é a seleção do conjunto ou amostra de dados mais apropriada. Inúmeras empresas utilizam pelo menos um sistema de gestão dos processos organizacionais. Como cada sistema normalmente adota um mecanismo próprio de geração dos *logs* de eventos, existe uma grande variedade de estruturas e formatos de armazenamento que o especialista em mineração deve compreender para utilizá-los adequadamente. Outro aspecto é que os SI são utilizados de diferentes formas, requerendo tempo de análise para levantar os dados relevantes. É necessário verificar como os dados poderão ser extraídos e tratados para as fases subsequentes do processo de mineração.

Portanto, não é raro que o especialista em mineração utilize um método empírico de tentativa e erro, ajustando ou revisando escolhas, sucessivamente, até atingir resultados que possam ser considerados coerentes e significativos. No entanto, esse processo pode se tornar bastante dispendioso em termos de tempo, exigindo do especialista considerável experiência na sequência de passos a serem executados e nos possíveis desdobramentos que possam ocorrer.

1.2 Objetivos

O objetivo geral deste trabalho é propor um método de MP que apoie a tomada de decisões dentro da organização, descrevendo os passos envolvidos, relatando ferramentas, algoritmos ou técnicas que podem ser utilizadas e alguns resultados que podem ser obtidos nesse processo investigativo. Em especial, será dado enfoque em ilustrar cada passo do método de uma forma mais didática e detalhada, para que seja mais intuitivo compreender a sua aplicação em casos reais. Para que o apoio à tomada de decisões seja possível, serão levantadas algumas perguntas iniciais estratégicas aos gestores envolvidos e, ao final, as informações descobertas serão utilizadas para responder essas perguntas.

O método será aplicado na análise do processo de controle de férias de servidores em um Tribunal Federal, utilizando *logs* de dados reais para responder as seguintes perguntas:

(1) É possível reduzir custos na execução dos passos no processo do controle de férias?

(2) Existe volume significativo de retrabalho?

(3) Como o sistema envolvido implementou o fluxo de controle de férias?

(4) Quais são os reais recursos e papéis envolvidos no controle de férias?

(5) Qual é o nível de envolvimento dos gerentes no processo?

(6) Existe sobrecarga de recursos no controle de férias?

1.3 Contribuições

Ao apresentar uma proposta de método de mineração para apoio na formação de idéias, resolução de problemas e tomada de decisões, é esperado que os técnicos de mineração possam ganhar maior entendimento nos típicos desafios e dificuldades que enfrentarão ao longo do trabalho de mineração. Não somente isso, mas que tenham orientação em possíveis opções e escolhas que possam tomar para melhor explorar os diversos algoritmos e técnicas existentes.

Além de apresentar um método, será dada ênfase na aplicação do método, descrevendo os passos envolvidos, considerações ou diretrizes, além de ferramentas que possam ser utilizadas. Através desse detalhamento, espera-se que o técnico de mineração possa entender melhor como um projeto com características similares possa ser realizado, e que seja possível maximizar os ganhos dos resultados para uma posterior revisão e otimização dos processos organizacionais analisados.

1.4 Descrição dos Capítulos

Neste trabalho é apresentada no Capítulo 2–Fundamentos uma breve revisão, incluindo conceitos e métodos relacionados ao trabalho; no Capítulo 3–Trabalhos Correlatos são descritos alguns trabalhos correlatos; no Capítulo 4–Apresentação do Método o método é apresentado e ilustrado; no Capítulo 5–Cenário de Aplicação e Resultados os resultados da aplicação do método são discutidos; e, finalmente, no Capítulo 6–Conclusões e Trabalhos Futuros as conclusões e os trabalhos futuros são expostos.

Capítulo 2

Fundamentos

Neste capítulo serão apresentadas as áreas relacionadas a esse trabalho. Na Seção 2.1 será apresentada uma visão geral da área de MP incluindo: propósitos, comparação com abordagens correlatas, conceitos, etapas, algoritmos, ferramentas e métodos. Na Seção 2.2 será descrito o processo de controle de férias, processo escolhido para validar o método proposto em um caso real.

2.1 Mineração de Processos

De acordo com o Manifesto de Mineração de Processos (van der Aalst et al., 2011), criado pelo grupo *IEEE Task Force on Process Mining*, existem três propósitos de utilização da MP:

1. Descoberta: lida com a geração de um modelo a partir dos *logs* de eventos, sem utilizar qualquer meta informação. Este processo descobre não só o fluxo de controle do sistema, mas também os modelos organizacionais.
2. Conformidade ou auditoria: lida com a comparação de modelos pré-existentes com o modelo minerado, com o objetivo de detectar inconsistências e/ou desvios.
3. Aprimoramento: tem como objetivo estender ou aprimorar o modelo de processo existente a partir de informações recuperadas do *log* de eventos.

Outros propósitos de análise são mencionados por Buijs (2010), como:

1. Desempenho: tem como objetivo permitir a visualização e análise do desempenho das atividades realizadas nos processos, que podem ser medidas através de indicadores de tempo, custo ou qualidade (Hornix, 2007).
2. Rede social: permite analisar a rede social envolvida com o processo, caracterizando, entre outros, a densidade da rede, as distâncias sociais entre os recursos, casos específicos de recursos "estrela" (ligados a vários outros) ou recursos "isolados" (não ligados aos outros) (van der Aalst and Song, 2004).
3. Regra de negócio: permite validar se alguma regra de negócio definida no processo está, de fato, sendo seguida.

4. Predição: tem como objetivo predizer o que pode acontecer em futuros casos baseado em dados históricos do *log* de eventos.

Esses propósitos tornam possível a descoberta, diagnóstico, monitoramento e melhoria de processos, ao extrair valioso conhecimento a partir de dados reais que são recuperados diretamente dos SI envolvidos (van der Aalst, 2016).

2.1.1 Conceitos

van der Aalst and van Hee (2012) descrevem alguns conceitos básicos aplicáveis a todos os processos no escopo de MP, a saber:

- Tarefa: é uma unidade lógica de trabalho que é realizada por um recurso. Não necessariamente o recurso executa integralmente a tarefa, mas ele é o responsável por ela.
- Recurso: é o nome genérico de uma pessoa, máquina, grupo de pessoas ou máquinas que são designadas tarefas para que as executem.
- Atividade: é a execução de uma tarefa por um recurso.
- *Case*: é o elemento que é produzido ou modificado num trabalho. Também pode ser referenciado como serviço, produto ou item. Cada *case* tem uma identidade única, além de ter um tempo de vida limitado.
- Processo de negócio: consiste de um número de tarefas e um conjunto de condições que determina a ordem das mesmas. Um processo também pode ser chamado de procedimento.
- Roteamento: é o caminho ao longo de ramos em particular de um processo, determinando quais tarefas devem ser realizadas, e em qual ordem, considerando que, no geral, pode existir mais de um caminho.

Além disso, evento foi definido por Fluxicon (2012)como uma atividade executada num processo.

2.1.2 Tipos de Processos

A classificação dos processos pode ser vista como uma escala de valores, ou um gradiente, sendo um extremo o tipo estruturado e o outro o não estruturado. Portanto, a maioria dos processos, na prática, está cituada em algum ponto entre esses dois extremos de classificação. O tipo estruturado, também denominado "processo lasanha", é regular, repetitivo e pode ser controlado. Isso significa que as instâncias do processo ocorrem numa frequência relativamente constante, têm aproximadamente a mesma duração de tempo e seguem um ritmo contínuo e regular. Além disso, o caminho principal do fluxo do processo normalmente pode ser representado por um modelo relativamente simples, e os usuários chaves do processo sabem confirmar a sua validade. Na Figura 2.1 pode ser visto um exemplo desse tipo de processo.

Já o processo não estruturado, também denominado "processo espaguete", é irregular, flexível e variável. O nome "espaguete" deriva do formato que o modelo do processo normalmente toma, com tantas conexões que fica muito difícil entender qual é a estrutura. Na Figura 2.2 pode ser visto um modelo de exemplo desse tipo de processo.

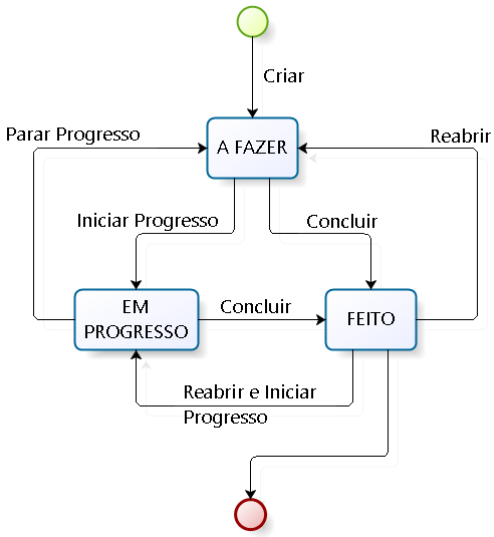

Figura 2.1: Exemplo de "processo lasanha".

2.1.3 Etapas ou Fases

A disciplina de MP pode ser subdivida em etapas ou fases com marcos e objetivos bem definidos e específicos, com o propósito de facilitar o entendimento e melhor controlar o progresso dos trabalhos. As seguintes etapas são consideradas aplicáveis à grande maioria dos projetos de MP:

- Iniciação: etapa de definição do escopo da MP. Nessa etapa será feito um levantamento de necessidades junto ao cliente requisitante, além de um estudo preliminar dos SI envolvidos. O principal objetivo é tentar atingir o melhor escopo teoricamente factível, porque ainda não se sabe se os dados disponíveis serão suficientes.
- Pré-análise dos dados: etapa de exploração dos dados gerados em *logs* de eventos pelos SI envolvidos, para entender como eles estão estruturados e formatados, quais dados são relevantes para o escopo definido, se são suficientes e completos, aplicar análise estatística e de padrões para melhor escolher as amostras dos *logs*, entre outros.
- Tratamento dos *logs*: mapeamento e conversão dos *logs* selecionados para um formato mais apropriado de mineração (e.g., *Mining eXtensible Markup Language* - MXML ou *eXtensible Event Stream* - XES).
- Mineração: mineração dos *logs* formatados para análise. O escopo da análise depende das necessidades iniciais, podendo ser realizada para levantar o modelo do fluxo de trabalho, validar a conformidade de um modelo previamente definido, analisar desempenho ou a rede social, entre outros.
- Avaliação e entrega dos resultados: avaliação dos resultados através de métricas aplicáveis para nivelar a qualidade alcançada, e devida entrega ao cliente requisitante para que ele os entenda.

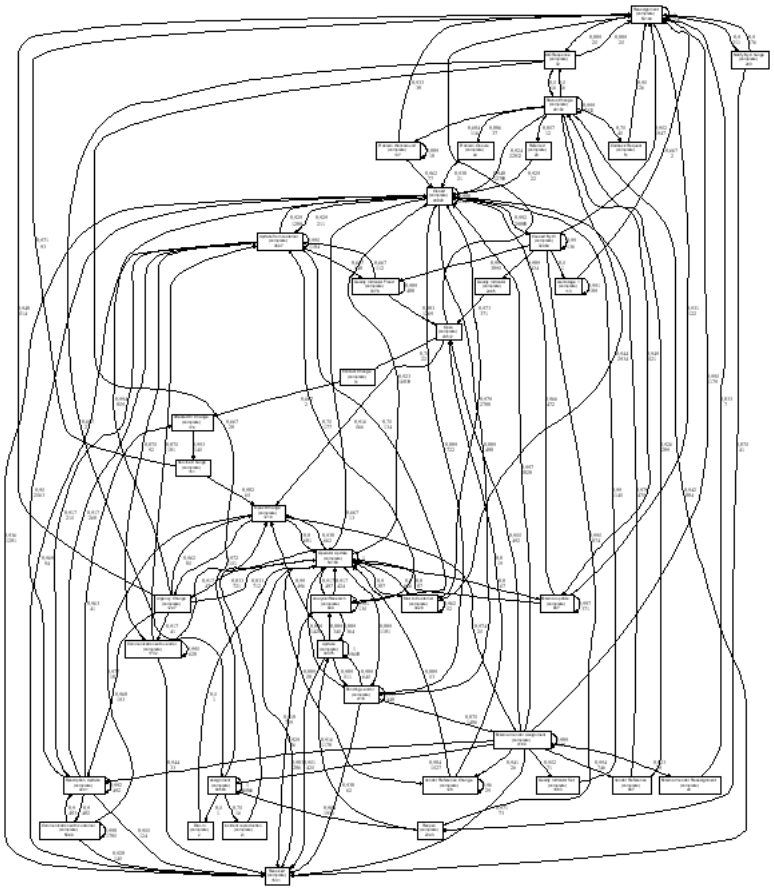

Figura 2.2: Exemplo de "processo espaguete".

Vários obstáculos e desafios incidem na disciplina de MP, em especial na etapa de pré-análise dos dados. Alguns, em particular, são mais comuns e frequentes, e foram esboçados por van der Aalst et al. (2011):

- Ruído: dados de *logs* armazenados podem estar incorretos ou incompletos, provocando problemas quando esses dados são minerados.
- Tarefas escondidas: tarefas que existem, mas não podem ser encontradas nos dados.
- Tarefas duplicadas: ocorrem quando dois nós de processos se referem ao mesmo modelo de processo.
- Fluxos de escolhas controladas: escolhas ao longo do processo que dependem de escolhas feitas em outras partes do modelo.
- *Loops* de mineração: *loops* em que um ou mais eventos são executados várias vezes.
- Processos concorrentes: processos que ocorrem ao mesmo tempo.
- Pesquisa local/global: estratégias locais restringem o espaço de busca, mas são menos complexas. Já estratégias globais são complicadas, mas tem uma melhor chance de encontrar a solução ótima.
- Perspectivas diferenciadas: eventos com informação adicional que permitem gerar diferentes perspectivas de mineração e análise dos dados.

2.1.4 Notações de Modelagem de Processos

van der Aalst (2011b) descreve algumas notações básicas existentes:

- Sistema de Transição: é uma das notações mais básicas, consistindo de estados e transições. A Figura 2.3 mostra um sistema de transição consistindo de oito estados. Ele modela o tratamento de retirada e devolução de livros numa biblioteca. Existe um estado inicial rotulado s1 e um estado final rotulado s4. Cada estado possui um único rótulo, que serve apenas de identificador e não possui nenhum significado. Cada transição conecta dois estados com um arco e possui um rótulo que representa o nome da tarefa correspondente.

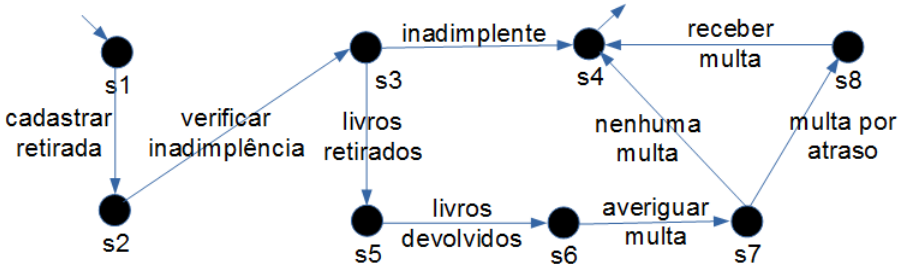

Figura 2.3: Exemplo de sistema de transição para a retirada e devolução de livros numa biblioteca.

- Rede de Petri: é uma das linguagens de modelagem de processos mais antiga, que permite modelar e analisar os processos através do formalismo, já que previne ambiguidades, incertezas e contradições (van der Aalst and van Hee, 2012). Possui um único ponto de entrada, um único de saída, e cada lugar (componente passivo) representa uma condição, e cada transição (componente ativo) uma tarefa. A estrutura da rede é fixa, mas os *tokens* podem fluir ao longo dela através de regras de acionamento. A Figura 2.4 mostra uma rede de Petri para o mesmo exemplo de uma biblioteca. Uma transição fica habilitada se todas as suas entradas possuem *tokens*. Ao executar ou acionar uma transição habilitada, um *token* de cada entrada é consumido e é produzido um *token* em cada saída.

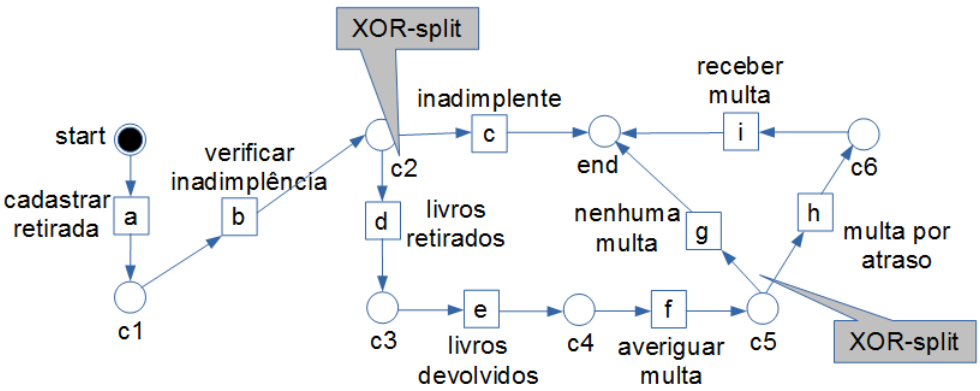

Figura 2.4: Exemplo de rede de Petri para a retirada e devolução de livros numa biblioteca.

- BPMN: é uma das linguagens mais utilizadas recentemente para modelar processos de negócio. Ela é suportada por muitos desenvolvedores de ferramentas de modelagem e foi padronizada pela *Object Management Group* - OMG[1]. Consiste basicamente de atividades atômicas chamadas tarefas, *gateways* que definem lógica de execução e eventos, parecidos com posições nas redes de Petri, mas com condições especiais de entrada e/ou saída. A Figura 2.5 mostra um modelo BPMN para o mesmo exemplo de uma biblioteca.

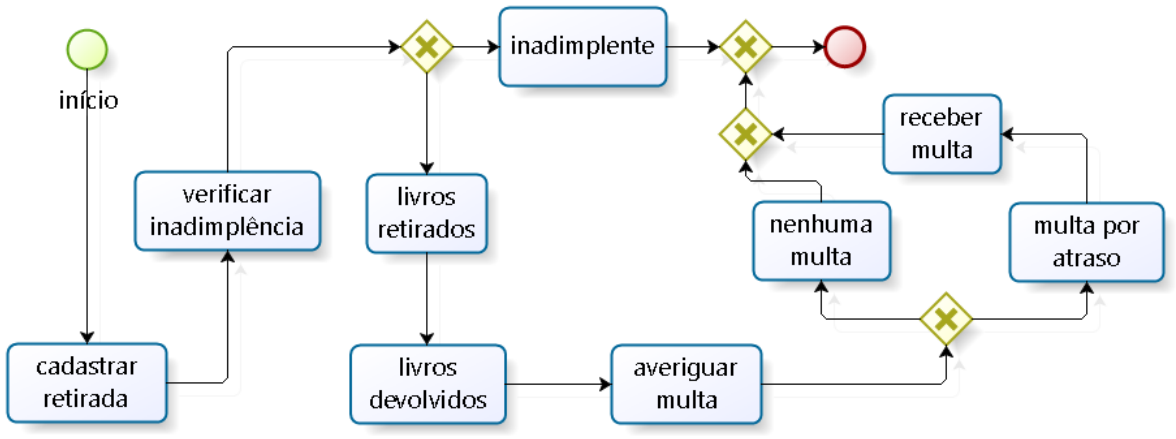

Figura 2.5: Exemplo de modelo BPMN para a retirada e devolução de livros numa biblioteca.

2.1.5 Algoritmos

A Tabela 2.1 apresenta um agrupamento dos algoritmos para descoberta do modelo do fluxo de acordo com Tiwari et al. (2008), além de possíveis aplicações desses algoritmos e exemplos. Dentre os algoritmos estudados estão o α *Miner* e β *Miner*, propostos por van der Aalst et al. (2004) e citados por vários autores por terem sido projetados para trabalhar com um conjunto de condições e estruturas de processos. Ambos os algoritmos mineram *logs* de eventos para fazer descoberta do modelo de fluxo de trabalho e representá-lo através de uma rede de Petri, sendo que o primeiro algoritmo parte das dependências locais de eventos e o segundo considera o elemento tempo.

O algoritmo *Fuzzy Miner* foi desenvolvido por Günther and van der Aalst (2007) após notarem que os primeiros algoritmos de mineração, dentre eles o α *Miner*, produziam modelos denominados "espaguete", considerando os processos pouco estruturados. Para tornar o modelo gerado mais compreensível, o algoritmo faz algumas simplificações através das seguintes considerações:

- Comportamento de maior importância é preservado;
- Comportamento de menor importância, mas altamente correlacionado, é agregado, ou seja, escondido em *clusters* no modelo simplificado;
- Comportamento de menor importância e pouco correlacionado é abstraído, ou seja, removido do modelo simplificado.

[1] http://www.omg.org/

Tabela 2.1: Algoritmos para descoberta do modelo do fluxo (adaptado de Tiwari et al. (2008, p. 6)).

Tipo	**Descrição**	**Exemplos**
Baseados em Mineração de Dados	Utilizam algoritmos de mineração de dados para extrair informações de processos	InWoLvE (Herbst and Karagiannis, 2004)
Genéticos	Algoritmos projetados em torno do processo de seleção natural de Darwin	Genetic Miner (van der Aalst et al., 2005)
De Abordagem Markoviana	Algoritmos que examinam o comportamento passado e futuro para definir um potencial estado corrente	Expectation-Maximization (Szimanski, 2013)
Análise em Clusters	Divide um grupo de soluções em subgrupos homogêneos	Sequence Clustering (Rebuge, 2012)
Rede Neural	Modelam o raciocínio humano e sua habilidade de "aprender" para então identificar padrões nos dados	RNet (Cook and Wolf, 1998)
Outras Abordagens	Algoritmos customizados, projetados para MP por autores individuais	α *Miner* e β *Miner* (van der Aalst et al., 2004) Fuzzy Miner (Günther and van der Aalst, 2007) Heuristics Miner (Weijters and de Medeiros, 2006)

Weijters and de Medeiros (2006) descrevem outro algoritmo muito conhecido, o *Heuristics Miner*, que inclui heurísticas para iniciar e efetuar uma sintonia fina na descoberta do processo através dos *logs*. A motivação de desenvolver esse algoritmo foi de criar uma técnica menos sensível a ruído e incompletude do *log* de eventos, considerando dependências causais de eventos e utilizando um grafo de dependências.

De acordo com Weijters and de Medeiros (2006), o ponto de partida desse algoritmo é a construção de um grafo de dependências. Para tanto, deve-se ter em mãos a frequência com que as tarefas ocorrem em sequência, para cada par de tarefas. Por exemplo, considerando as tarefas **a** e **b**, a notação $|a >_w b|$ representa essa frequência, ou seja, o número de *traces* em que b ocorre logo após a. Em seguida deve ser computada uma tabela de dependências através da fórmula:

$$a \Rightarrow_w b = \left(\frac{|a >_w b| - |b >_w a|}{|a >_w b| - |b >_w a| + 1} \right)$$

Posteriormente, deve ser identificada na tabela a tarefa de início do fluxo, que corresponderá à linha contendo apenas valores positivos, e a tarefa de término do fluxo, contendo apenas valores negativos. Quando a heurística *All tasks connected* estiver acionada, inicia-se na tarefa de início do fluxo, procurando o maior valor de outra célula existente na mesma linha. Caso existam dois ou mais valores iguais, um deles é escolhido arbitrariamente. A tarefa correspondente à coluna desse valor servirá de início para a busca seguinte pelo maior valor da linha correspondente, continuando assim de forma recursiva até alcançar a tarefa de término do fluxo. As células contendo os maiores valores das linhas determinam o fluxo principal do processo. Cada célula de maior valor define um arco de conexão entre a tarefa origem, correspondente à linha da célula, e a tarefa destino, correspondente à coluna da célula. Além disso, outras células de valores positivos na tabela que atendam a alguns critérios, dependendo dos *thresholds* definidos, também serão consideradas para gerar arcos adicionais.

Tanto o algoritmo α *Miner*, quanto o *Fuzzy* e *Heuristics* estão disponíveis como plugins da ferramenta ProM[2] (van Dongen et al., 2005). A ferramenta Disco[3], por outro lado, adota um algoritmo que pode ser considerado uma versão melhorada do *Fuzzy Miner* disponível no ProM (van der Aalst, 2016).

2.1.6 Ferramentas

Dentre as ferramentas existentes, três em especial foram escolhidas para estudo e utilização. O ProM foi escolhido porque van der Aalst (2016) o descreve como a ferramenta líder de código aberto na área de MP, com um total de 1.500 *plug-ins* disponíveis (incluindo os obsoletos), desenvolvida e mantida por vários grupos de pesquisa de diversos países ao redor do mundo, e baixada por dezenas de milhares de organizações, com mais de 130.000 *downloads* realizados. Por causa do sucesso do ProM na comunidade acadêmica, existem poucas ferramentas adicionais que sejam gratuitas. Uma delas é o PMLAB[4], um ambiente de *script* para MP desenvolvido em Barcelona e inspirado nas ferramentas MATLAB e Mathematica. Ele é capaz de ler arquivos XES, MXML ou CSV e encadear diferentes passos de análise, utilizando a teoria de regiões para aplicar técnicas de descoberta de processos (van der Aalst, 2016). Outra delas é o RapidProM[5], uma extensão da ferramenta RapidMiner (uma popular ferramenta de mineração de dados) que permite encadear blocos de construção para executar fluxos de análise utilizando *plug-ins* do ProM de forma nativa.

A ferramenta comercial Disco foi escolhida para porque é frequentemente utilizada por pesquisadores para realizar a análise inicial dos *logs* através da filtragem, exploração e análise de gargalos (van der Aalst, 2016). Outras ferramentas comerciais, como Celonis Process Mining[6] (Celonis), Enterprise Discovery Suite[7] (EDS), Interstage Business Pro-

[2] http://www.promtools.org/
[3] http://fluxicon.com/disco/
[4] http://www.cs.upc.edu/~jcarmona/PMLAB/
[5] http://www.rapidprom.org/
[6] http://www.celonis.de/
[7] http://www.stereologic.com/

cess Manager Analytics[8] (Fujitsu) e Minit[9] (Minit), não foram incluídas no escopo desta pesquisa.

A ferramenta XESame foi escolhida para dar apoio ao ProM e Disco, já que permite gerar *logs* no formato XES ou MXML a partir de dados armazenados num banco de dados relacional.

As ferramentas escolhidas estão descritas a seguir:

- ProM: é uma ferramenta gratuita e de código-fonte aberto, consistindo de um *framework* que permite acoplar *plug-ins* desenvolvidos por terceiros para diferentes finalidades, dentre elas: visualização gráfica de processos, algoritmos de mineração e ferramentas de análise. O *framework* incentiva uma abordagem única de arquitetura, de tal forma que os usuários utilizem uma interface genérica para operar todo o conjunto de algoritmos de modelagem e MP.
- Disco: é uma ferramenta comercial que vem ganhando destaque. Uma característica relevante é que não há necessidade de decidir qual algoritmo de mineração deve ser usado, simplificando o uso e eliminando a diversidade de diferentes notações de modelos gerados pela mineração.
- XESame (Buijs, 2010): é uma ferrramenta gratuita e de código-fonte aberto, que serve para mapear e converter *logs* de eventos gerados pelos SI, a partir de um banco de dados relacional, para um formato mais apropriado para a MP, especificamente para XES[10] e MXML[11].

2.1.7 Métodos e Modelos

Bozkaya et al. (2009) propõem um método de MP que, teoricamente, pode ser aplicado em qualquer caso de uso ou domínio de processos, mostrado na Figura 2.6 através da notação BPMN. Um ponto chave desse método é a dispensa de experiência prévia ou conhecimento específico do negócio, a única informação necessária é o *log* de eventos. O método proposto consiste de seis fases, a saber:

1. Preparar o *Log*: o *log* de eventos é extraído do SI;
2. Inspecionar o *Log*: para ganhar entendimento inicial do processo;
3. Analisar o Fluxo de Controle: para conseguir visualizá-lo de uma forma ilustrativa e amigável;
4. Analisar Desempenho: para que, por exemplo, os possíveis gargalos sejam identificados;
5. Analisar Papéis: para identificar os recursos que realizam as tarefas e se trabalham em conjunto;
6. Transferir Resultados: repassando-os apropriadamente ao cliente para que ele consiga obter um melhor entendimento do SI envolvido, e assim torná-lo mais eficiente ou ágil quando aplicável.

[8] http://www.fujitsu.com/
[9] http://www.minitlabs.com/
[10] http://www.xes-standard.org/
[11] http://www.processmining.org/logs/mxml

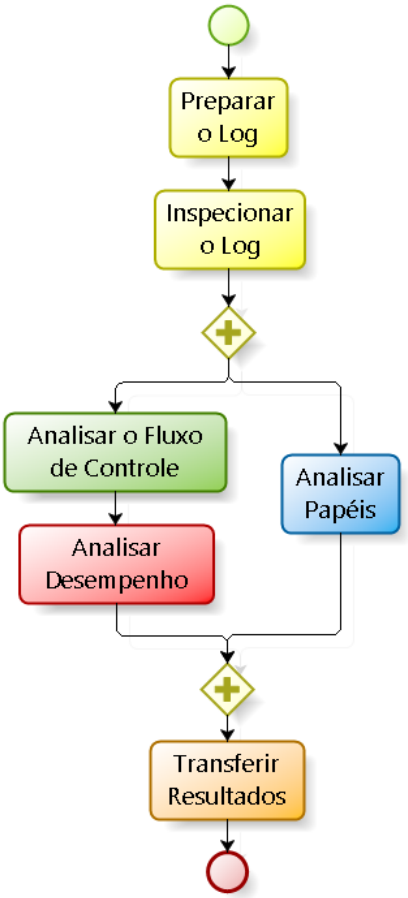

Figura 2.6: Método de Bozkaya na notação BPMN.

van der Aalst (2011a,b) propõe um modelo denominado Ciclo de Vida L*. Esse modelo descreve os estágios de um projeto de MP, em especial para processos estruturados, e está mostrado na Figura 2.7 através da notação BPMN. Os diferentes estágios estão descritos a seguir:

- Estágio 0: para planejar e justificar o projeto. Os projetos são caracterizados como sendo: (a) dirigidos pela disponibilidade dos dados, (b) dirigidos a responder determinadas dúvidas a respeito do processo, ou (c) dirigidos a atingir um objetivo qualquer, como suprir uma determinada métrica negocial;
- Estágio 1: para localizar, extrair e transformar os dados de eventos disponíveis. Esse estágio é caracterizado como não trivial, envolvendo vários aspectos e dificuldades. Além disso, envolve não apenas coletar dados de eventos, mas também modelos existentes e outros artefatos relacionados, lembrando de explorar ao máximo o conhecimento existente, principalmente dos técnicos envolvidos;
- Estágio 2: para criar um modelo do fluxo do processo e conectá-lo ao *log* de eventos. Esse é um passo extremamente iterativo, ou seja, deve ser executado várias vezes, até alcançar um resultado considerado aceitável;
- Estágio 3: para agregar outras perspectivas ao modelo, como tempo envolvido e recursos;

- Estágio 4: para prover suporte operacional. Neste momento, não se dedica apenas tempo para dados históricos, mas para utilizar MP em tempo real, permitindo, por exemplo, detectar desvios ou fazer predições do tempo remanescente do fluxo em execução. É importante observar que o estágio 4 só é possível nos processos estruturados.

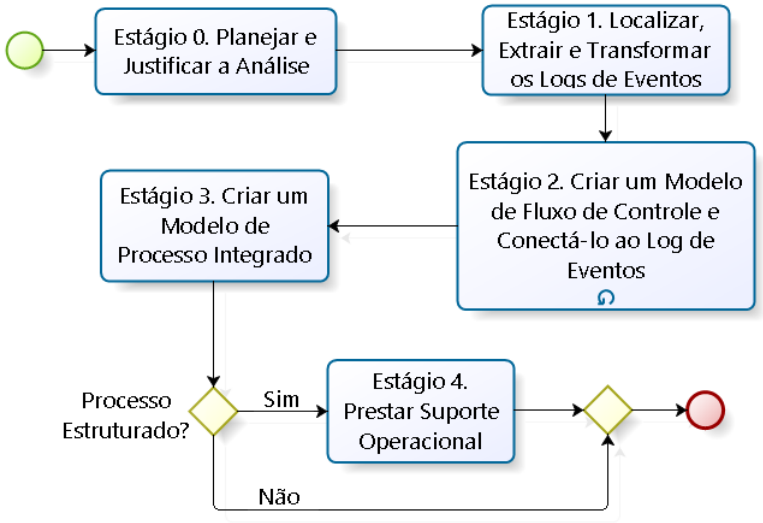

Figura 2.7: Modelo de Ciclo de Vida L* de van der Aalst (2011b) na notação BPMN.

Apesar do método de Bozkaya et al. (2009) definir uma sequência de passos alto nível de um projeto de MP, possibilita o especialista em MP orientar-se sobre os passos que deverão ser realizados, independente do contexto negocial que seja adotado. O método proposto por van der Aalst (2011a,b) possui um caráter mais abrangente, permitindo que a MP seja utilizada não só para descoberta, exploração e diagnóstico de processos, como também para prestar suporte operacional a esses processos em tempo real. Portanto, contempla uma abordagem mais completa de aplicação da MP no ciclo de vida dos processos. A escolha do método adotado neste trabalho será descrita mais adiante, no Capítulo 4.

2.1.8 Outras Abordagens

Para gerir negócios de uma forma mais eficiente, acaba se tornando uma necessidade comum entre os administradores ou gestores analisar e cruzar informações relevantes dos diferentes setores da organização. O uso da Inteligência de Negócios (*Business Intelligence* - BI) permite que as organizações possam realizar uma série de análises e projeções para auxílio efetivo à tomada de decisão. A aplicação de BI não fica restrita a essa finalidade, sendo também bastante útil na otimização de trabalhos, redução de custos e melhoria nas estratégias do negócio.

No entanto, as típicas técnicas de análise em BI não são orientadas a processos, apesar de haverem abordagens integradoras (BI orientada a processos). Isso significa que alguns problemas ou melhorias podem ser mais difíceis de identificar quando não analisados na ótica de um fluxo completo de trabalho. Ou, ao serem identificados, não ficam

devidamente caracterizados dentro do contexto do processo relacionado, dificultando o entendimento de como podem ser tratados.

Por outro lado, a Modelagem de Processos de Negócio (*Business Process Modeling* - BPM) é uma atividade relativamente antiga (data de 1920), sendo a mesma uma importante técnica de análise orientada a processos. Através de BPM é possível visualizar os fluxos de trabalho numa representação gráfica para posterior análise e melhoria. Existem várias formas de notação visual dos modelos, incluindo fluxogramas, diagramas de fluxo de dados ou de blocos funcionais.

Mesmo sendo utilizada nas organizações, e também de grande importância, as típicas técnicas de análise através da BPM apresentam algumas deficiências. Uma delas é a de que o modelo desenhado do fluxo não necessariamente representa a realidade. É importante lembrar que, normalmente, os processos possuem grande dinamicidade, passando por mudanças que podem não estar refletidas no modelo, além de possíveis desvios na execução, decorrentes de falhas no planejamento inicial do fluxo, falta de conhecimento na realização dos trabalhos ou até mesmo fraudes. Outra deficiência é a de que modelos de processos por si só retratam a realidade de uma forma limitada, onde a informação predominante é a sequência de atividades e, eventualmente, os recursos ou departamentos envolvidos. No entanto, várias informações adicionais podem ser úteis durante a análise: tempo de execução das atividades, tempo de espera entre atividades, recursos efetivamente envolvidos, frequências que os caminhos no fluxo foram tomados (considerando que o fluxo tem mais de um caminho possível), dados que não são relacionados nesse tipo de modelagem.

É por isso que uma técnica mais completa e orientada a processos se torna crucial. A MP funciona como uma ponte que une as duas visões, conforme ilustrado na Figura 2.8: análise e cruzamento de informações úteis para a tomada de decisões (BI) a partir de um modelo completo do processo, incluindo não só a representação visual do mesmo (BPM), como também as perspectivas de tempo, recursos, custos, regras e dados. A prática de MP deriva da mineração de dados, mas ao mesmo tempo apresenta características próprias que as diferenciam da primeira.

2.2 Processo de Controle de Férias

O processo de controle de férias compreende os passos necessários para permitir que o servidor possa gozar das férias às quais tem direito, envolvendo atividades que englobam, desde o planejamento e concessão, até o efetivo pagamento das férias pelo empregador e gozo pelo empregado ou funcionário.

Para o regime de contratação CLT, o decreto-lei 5.452 de 1943[12] aprova a Consolidação das Leis de Trabalho, a qual dispõe que todo empregado que cumpre jornada normal de trabalho (8 horas diárias) tem direito a um período de férias de trinta dias corridos a cada doze meses de vigência do contrato de trabalho, com exceção dos casos em que tenha faltado ao serviço mais de cinco vezes. Nesses casos, é feito um abatimento desse período de descanso.

Já no caso de regime estatutário, próprio da administração pública direta, as leis aplicáveis podem ser tanto federais, estaduais quanto municipais. A Lei Federal 8.112,

[12] http://www.planalto.gov.br/ccivil_03/decreto-lei/del5452.htm

do regime jurídico dos servidores públicos[13], dispõe que todo servidor tem direito a um período de férias de trinta dias, parcelado em até três períodos, a cada doze meses de vigência do contrato de trabalho, sem nenhum tipo de abatimento por faltas. Em caso de parcelamento, o pagamento das férias ocorrerá sempre no primeiro período.

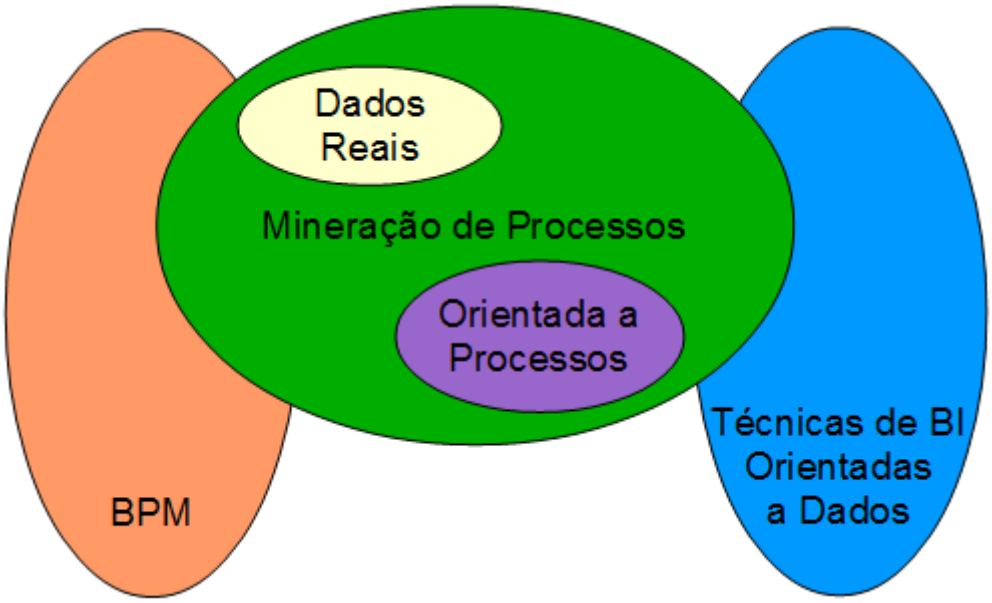

Figura 2.8: MP como uma ponte entre as visões de BPM e BI.

Esse conjunto de leis define questões gerais e abrangentes a respeito do direito do trabalho. No entanto, não contemplam todos os casos ou situações que ocorrem nas organizações, gerando algumas lacunas jurídicas. Para evitar esse problema, essa legislação também permite a definição de regulamentos ou regimentos internos complementares. O regulamento interno é um instrumento pelo qual o empregador pode estabelecer regras em termos de direitos e obrigações aos seus empregados, contanto que não contrariem as leis relacionadas, convenções, acordos coletivos e decisões das autoridades competentes.

No caso do Tribunal Federal de Justiça, que servirá de caso real de aplicação do método proposto neste trabalho, além das leis aplicáveis no regime estatutário, estão também as leis definidas no regimento interno do órgão, que estabelecem algumas regras específicas de férias. Dentre elas, estabelece que os desembargadores e juízes têm direito a um período de férias de sessenta dias anuais, parcelado em até duas vezes.

Com relação ao sistema de gestão de pessoas utilizado nesse Tribunal Federal, o planejamento e concessão de férias inicia alguns meses antes do início do ano de exercício, nos meses de Outubro e Novembro. A concessão é feita, ou através da funcionalidade de geração da escala, que cria todas as concessões de uma só vez, ou manualmente, uma a uma. A concessão é uma permissão emitida pelo setor de Recursos Humanos (RH) que permite o agendamento e gozo das férias por um servidor. Uma vez criada a concessão, o servidor é notificado e acessa o portal do sistema para marcar as férias. Após a marcação, o gestor é notificado para que possa entrar no mesmo portal e homologá-las ou rejeitá-las. Posteriormente, caso o servidor deseje remarcá-las, solicitará essa alteração ao setor de

[13] http://www.planalto.gov.br/ccivil_03/leis/L8112cons.htm

RH, já que apenas somente ele possui acesso para efetuar essa operação. O fluxo alto nível do processo de férias implementado por esse sistema está representado na Figura 2.9.

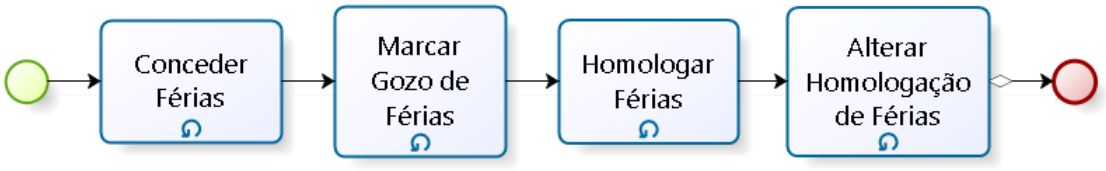

Figura 2.9: Processo de férias no Tribunal Federal.

Com relação à marcação de férias, o servidor pode dividir as férias em até três períodos, cada um com, no mínimo, dez dias. A exceção à essa regra ocorre no último período, em que o servidor necessariamente deve liquidar o saldo restante, seja ele maior ou menor que dez dias. A partir do segundo ano de trabalho, o servidor pode marcar o período de férias em qualquer dia do ano, não requerendo um período mínimo de espera em comparação com as férias gozadas no ano anterior. Como a primeira marcação de férias do ano envolve o pagamento de direitos e benefícios, deve ser marcada com um período mínimo de trinta dias de antecedência.

Com relação à homologação de férias, existe uma preocupação comum entre os gestores e coordenadores de evitar que as área e setores de suas responsabilidades se mantenham com um mínimo de profissionais atuantes para atender ao fluxo de demandas diário. Por isso, as férias são homologadas de tal forma que se evitem colisões significativas de períodos entre os diversos recursos da mesma localidade.

É importante também destacar que alguns fluxos excepcionais relevantes que ocorrem durante o período de gozo, como suspensão e interrupção de férias, não foram implementados por esse sistema. Essa situação torna o processo mais lento e burocrático, já que requer o uso de formulário em papel para efetuar as solicitações correspondentes. Além disso, outras etapas posteriores, como integração com a folha para pagamento das férias, não foram previstas no escopo deste trabalho.

Neste capítulo foram apresentados os fundamentos de MP relacionados com este trabalho, em especial os algoritmos de descoberta de processos α *Miner*, *Fuzzy Miner* e *Heuristics Miner*, as ferramentas ProM e Disco, e os métodos Ciclo de Vida L* e o proposto por Bozkaya et al. (2009). No Capítulo 3 será descrita a metodologia de pesquisa e três artigos correlatos a este trabalho de grande relevância.

Capítulo 3

Trabalhos Correlatos

A metodologia adotada neste trabalho de pesquisa foi investigativa e exploratória, iniciando com uma revisão do estado da arte dos algoritmos, técnicas e ferramentas, com especial enfoque nos métodos existentes de MP.

A revisão do estado da arte foi realizada através do método de revisão *quasi*-sistemática, conforme definido por Kitchenham (2004), através da ferramenta StArt (Zamboni et al., 2010), considerada uma das melhores existentes para essa finalidade (Marshall et al., 2014).

Uma revisão *quasi*-sistemática é uma revisão preliminar sobre determinado assunto de pesquisa, um meio de avaliar e interpretar toda pesquisa disponível e relevante a uma questão de pesquisa em particular, área de pesquisa ou fenômeno de interesse. Ela visa apresentar uma clara avaliação de um tópico de pesquisa através do uso de uma metodologia confiável, rigorosa e auditável. A condução de uma revisão sistemática é normalmente realizada em três fases: planejamento, execução e sumarização ou análise de resultados.

Durante a fase de planejamento foi elaborado um protocolo de pesquisa, consistindo dos seguintes itens:

- Questão de pesquisa: como definir um método de MP que dê apoio à otimização de processos e tomada de decisões dentro da organização?
- População: pesquisas em métodos de MP
- Resultados:
  - Identificação dos desafios e obstáculos durante a elaboração deste tipo de método
  - Proposta de um método que auxilie os especialistas na análise de processos através da MP
  - Avaliação da aplicabilidade de um método de MP no âmbito do controle de férias nas organizações
- Aplicação: apoiar a realização de projetos de MP

Durante a fase de execução, foram pesquisados 284 artigos publicados no IEEE, sendo que a Figura 3.1 ilustra a quantidade de artigos por ano. Foram identificados doze artigos

do IEEE contendo uma descrição mais detalhada do método utilizado para efetuar a análise dos processos. Além disso, os seguintes trabalhos foram selecionados como correlatos a este, e serão descritos a seguir: uma dissertação de mestrado da Universidade Técnica de Lisboa de 2012, um artigo publicado no Simpósio Brasileiro de Sistemas de Informação (SBSI) 2011 e o vencedor do desafio *Business Process Intelligence Challenge* (BPIC) 2015.

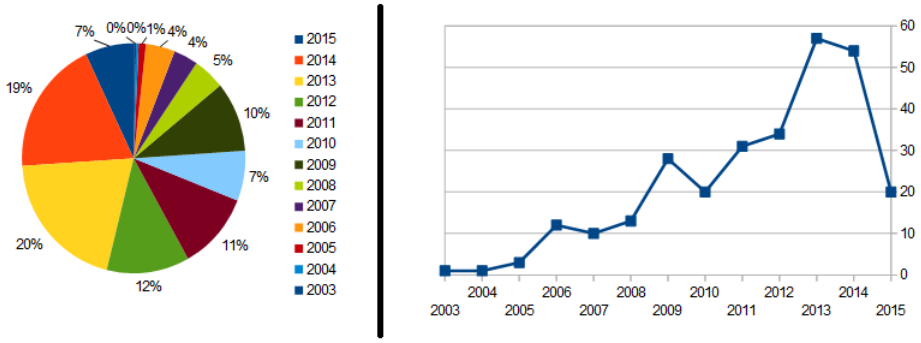

Figura 3.1: Artigos lidos na revisão *quasi*-sistemática, quantificados por ano.

Rebuge (2012)

Esta dissertação propõe um método de mineração que possa ser aplicado em ambientes da área da saúde, já que os processos dessa área possuem características especiais que requerem um cuidado diferenciado. Dentre essas características, as seguintes são citadas: altamente dinâmicos, extremamente complexos, cada vez mais multi-disciplinares e de caráter específico e pontual.

Para lidar com esse perfil de processo negocial, o trabalho propõe uma metodologia específica de mineração. Essa metodologia é validada num caso real, mais especificamente no Hospital de São Sebastião, localizado na cidade de Santa Maria da Feira, ao norte de Portugal. Para dar apoio à implementação da metodologia, foi desenvolvida uma ferramenta customizada para integrá-la com o SI envolvido nos processos analisados.

A metodologia proposta é uma estensão do estudo feito por Bozkaya et al. (2009). De acordo com a Figura 3.2, é possível encontrar pequenas diferenças em comparação com a metodologia original (vide Figura 2.6). A diferença mais significativa é a inclusão de um novo passo, o subprocesso *Sequence Clustering Analysis*, ilustrado na Figura 3.3. Esse subprocesso prevê a utilização do algoritmo *Sequence Clustering* nos *logs* de eventos. Nos passos seguintes, são realizadas algumas análises para selecionar os *clusters* mais adequados para utilizar no restante do método de mineração.

Para validação dessa proposta, foi desenvolvida uma nova ferramenta de MP que implementa a metodologia definida, denominada *Medtrix Process Mining Studio* (MPMS). O sistema ao qual essa ferramenta foi integrada chama-se Medtrix, desenvolvido internamente no hospital para fornecer aos médicos uma perspectiva integrada de toda a informação clínica relacionada com os pacientes, desde a sua admissão no hospital. Os dados desse sistema foram convertidos para uma nova base de dados simplificada, por causa da complexidade envolvida. A partir daí, a ferramenta oferece funcionalidades para filtrar os dados desejados para análise, aplicar o algoritmo de clusterização, analisar os resultados e, finalmente, aplicar algoritmos de mineração para fazer análise de desempenho e rede

social, tanto quanto exportar os *logs* tratados no formato MXML para posterior análise na ferramenta ProM.

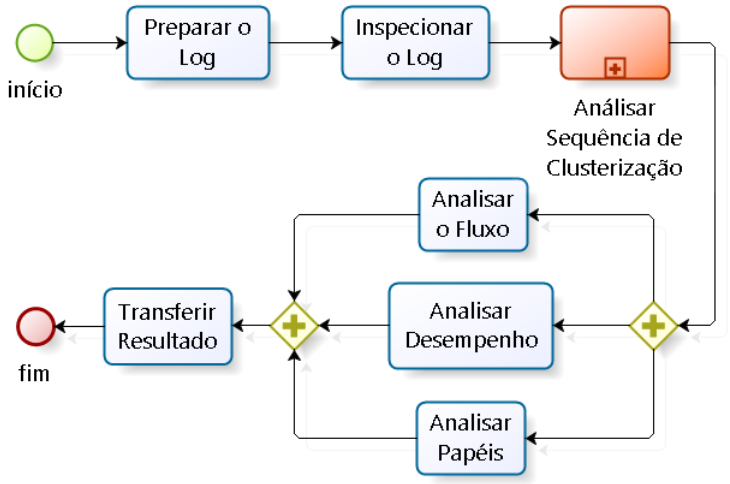

Figura 3.2: Metodologia de Bozkaya adaptada do método de Rebuge (2012).

O método propõe, em especial, a *clusterização* dos *logs*, para separá-los em grupos que tenham maior proximidade na execução do fluxo de atendimento. Essa técnica permite subdividir um processo que possua várias ramificações em subprocessos, e assim analisá-los agrupados por proximidade, diminuindo a complexidade geral envolvida.

O estudo de Rebuge (2012), assim como a metodologia original que foi estendida, possui relação com este ao propor um método de MP, com uma descrição das fases envolvida, algoritmos e técnicas utilizadas ao longo do processo.

van der Ham (2015)

O grupo IEEE Task Force organiza anualmente, desde 2011, um desafio de MP denominado BPIC, o qual é ligado ao *Workshop on Business Process Intelligence* (WBPI), realizado dentro da *International Conference on Business Process Management* (BPM). A Conferência BPM é uma das principais conferências da área de Gestão de Processos de Negócio (Capes Qualis A2 na C. Computação). O objetivo do desafio é oferecer uma oportunidade de realizar análise de MP em dados reais, tanto para praticantes quanto pesquisadores. São disponibilizados aos participantes *logs* reais e informações relevantes sobre a organização ou organizações envolvidas, a área de atuação dessa(s) organização(ões) e outras informações relacionadas. Além disso, são disponibilizadas algumas perguntas relativas ao processo que o participante deverá responder, sendo que é incentivado o uso de qualquer técnica ou ferramenta de mineração possível. Ao final, um juri escolhe o trabalho vencedor, e o autor envolvido recebe uma premiação pelo resultado alcançado.

No ano de 2015 (BPIC 2015), o desafio consistiu em analisar *logs* de cinco municípios da Holanda relativos às licenças de construção (construção de prédios, demolição de prédios, posicionamento de objetos numa via pública, derrubada de árvores, poluição ambiental, entre outros) de um período de quatro anos. Os *logs* contêm informações dos vários estágios do fluxo principal de requerimento da licença, além de possíveis objeções que

possam ter ocorrido. Podem existir diferenças de procedimentos entre os cinco municípios, e mudanças ao longo do tempo na legislação, regulamentações e procedimentos aplicáveis.

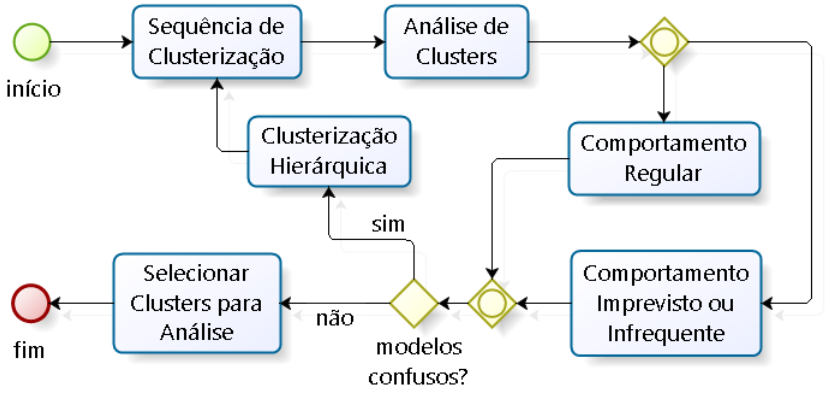

Figura 3.3: A metodologia *Sequence Clustering Analysis* (Rebuge, 2012).

O vencedor do BPIC 2015 foi o holandês Ube van der Ham, que definiu um método próprio de MP. O método definido não foi formalmente descrito nos documentos do BPIC, mas foi extraído por engenharia reversa e está ilustrado utilizando BPMN na Figura 3.4.

O Passo 1 do método foi dedicado a analisar o contexto do processo, considerando aspectos como: tipo de serviço prestado, objetivos, legislação aplicável, estrutura física, mudanças que ocorreram ao longo do período analisado.

Além disso, foi definida a abordagem de comparação dos processos nos Passos 2 e 4. Como estão sendo analisados ao todo cinco municípios, foram definidos indicadores de comparação, como velocidade de atendimento, aderência às leis / regulamentações e qualidade dos processos, que estejam relacionados com o desempenho dos municípios.

Em seguida, foram realizadas análises superficiais dos dados, além de filtragem e seleção dos eventos considerados relevantes no Passo 3. Algumas estatísticas básicas dos processos foram geradas, como tempo médio de atendimento, tempo médio por tipo de solicitação e tempo médio por ano.

Posteriormente, foram feitas análises relacionadas com o fluxo de controle, incluindo análise do volume de retrabalho (Passo 5), quantidades e frequências das variantes do fluxo (Passo 6) e descoberta do modelo do fluxo (Passo 7).

O Passo 8 consistiu em efetuar análises de desempenho, considerando diversas perspectivas de tempo, como o tempo médio dedicado para atender às solicitações, tempo total dedicado por atividade e recurso, gargalos no fluxo, entre outros. No Passo 9 foram analisadas mudanças no fluxo ao longo do tempo.

Em seguida, foram realizadas análises nos recursos envolvidos (Passo 10), identificando possíveis hierarquia de papéis (Passo 11), envolvimento dos papéis nas diferentes atividades (Passo 12), sobrecarga de recursos e volume de transferência de trabalho (Passo 13), especialização e compartilhamento de conhecimento (Passo 14).

No Passo 15 foram realizadas análises adicionais específicas do contexto do processo e, por fim, no Passo 16 foram apresentadas as conclusões e algumas recomendações finais. Uma das descobertas desse trabalho foi o excessivo consumo de tempo dedicado com objeções e recursos judiciais nas municipalidades, decorrentes de divergências de opiniões quanto às resoluções tomadas nos processos entre os requerentes e as municipalidades,

gerando desperdícios de tempo e dinheiro, e possível perda de credibilidade na qualidade dos trabalhos prestados. Além disso, foram ressaltadas diferenças significativas nos processos entre os municípios, em alguns casos gerando desdobramentos custosos no fluxo, apesar de terem sido projetados e implantados como um só, por um governo central.

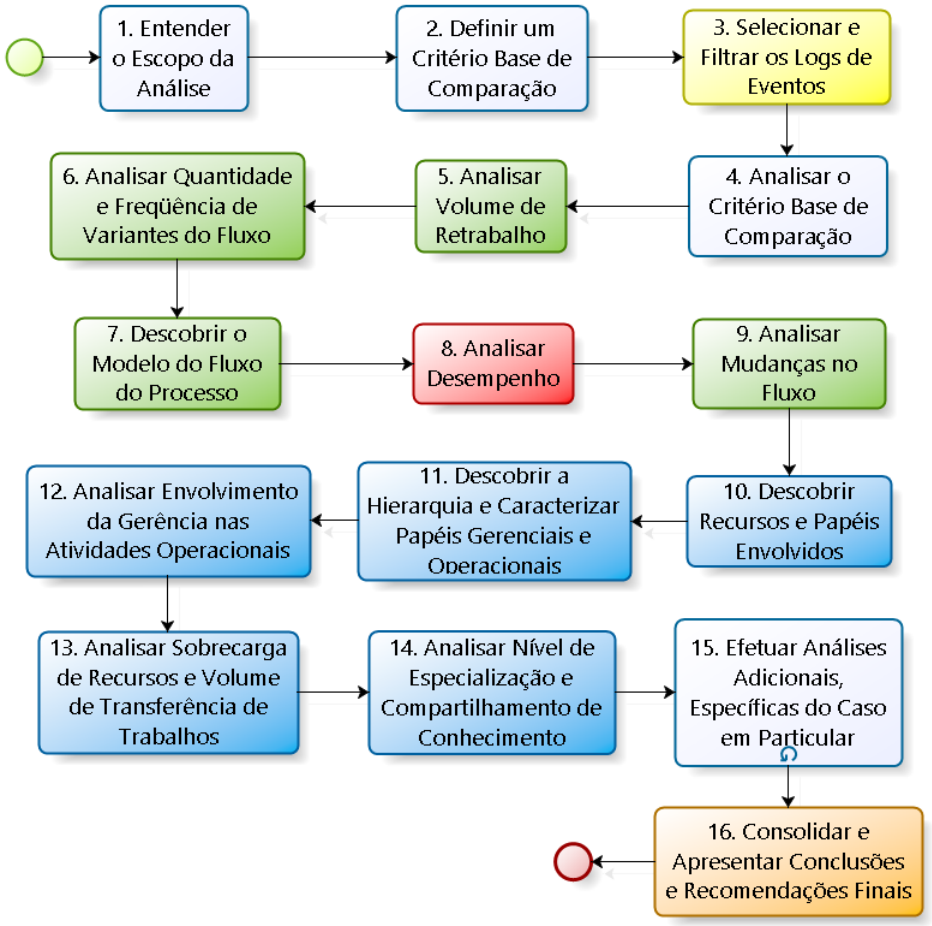

Figura 3.4: Engenharia reversa do método de van der Ham (2015) na notação BPMN.

O método utilizado por Ube van der Ham compreendeu a comparação dos processos de diferentes municípios. Uma técnica utilizada para fazer essa comparação foi a utilização de Cubos de Processos, muito útil na análise de processos mais complexos ou na comparação de diferentes processos, conforme descrito por van der Aalst (2013a). Essa técnica faz referência ao já conhecido Cubo de Dados OLAP (*Online Analytical Processing*), incluindo as operações associadas de fatiar (*slice*), cortar um subcubo (*dice*), aumentar (*drill-down*) ou diminuir (*drill-up*) o nível de detalhamento, sendo que, neste caso, a unidade básica de informação é o evento do processo. Dessa forma, é possível avaliar as várias dimensões de um ou mais processos de uma forma mais prática e flexível.

Francisco and Santos (2011)

Os autores propõem um método de mineração que permite extrair informações relevantes dos *logs* de processos como prática da gestão de conhecimento, para uma tomada de

decisões mais ágil e direta dentro das organizações.

O método foi validado numa aplicação real chamada Sistema de Gerenciamento de Anomalias (SGA), que tem por objetivo cadastrar, verificar e prover soluções para desvios nos processos de desenvolvimento de produtos, num sistema de gestão da qualidade de uma empresa do setor metal mecânico instalada em Joinville-SC. Os dados foram extraídos a partir de um banco de dados Microsoft SQL Server, filtrados e depois convertidos para o formato MXML. Posteriormente, o ProM foi utilizado para minerar os eventos do processo através de alguns algoritmos: α *Miner*, *Organizational Miner* e *Social Network Miner*. Finalmente, os resultados foram analisados com o objetivo de gerar conhecimento útil para a tomada de decisões.

O estudo de Francisco and Santos (2011) possui relação com esta pesquisa, já que apresenta e explora um método completo de mineração, descrevendo algoritmos e ferramentas utilizadas, com o objetivo de praticar a gestão de conhecimento e tomada de decisões no âmbito organizacional.

Neste capítulo foram apresentados os trabalhos correlatos de Rebuge (2012), van der Ham (2015) e Francisco and Santos (2011). No próximo capítulo será apresentado o método de MP proposto para dar apoio à tomada de decisões e otimizar os processos analisados.

Capítulo 4

Apresentação do Método

Neste capítulo será apresentado o método de MP proposto, incluindo os passos envolvidos, técnicas aplicáveis, dificuldades ou obstáculos, algoritmos e ferramentas que podem ser utilizadas. Várias ilustrações e exemplos serão providos para melhor contextualizar a aplicação do método.

Com o objetivo de aplicar a MP no apoio à tomada de decisões dentro de uma organização, primeiramente foi escolhido um método de referência que descreva os passos a serem realizados. O método escolhido foi o proposto por Bozkaya et al. (2009), pela sua simplicidade e flexibilidade, além de se propor aplicável a qualquer caso, conforme descrito na Seção 2.1.7.

Ao aplicar esse método na prática, foi necessário traduzir os passos abstratos e de alto nível que o compõem em atividades mais concretas e específicas. Portanto, para dar maior especificidade nesse método, foi feita a análise de alguns trabalhos práticos de mineração, em especial o método utilizado por Ube van der Ham, vencedor do desafio BPIC 2015. Apesar do método ser bastante específico, muitas atividades podem ser aplicadas em outros casos e inclusive ser relacionadas com os passos do método de Bozkaya et al. (2009). Essa comparação e alinhamento entre os dois métodos foi realizada e está representada na Figura 4.1, através da notação BPMN. Foi utilizada uma legenda de cores nas atividades para relacionar os passos dos dois métodos.

Para que essa junção fosse possível, algumas adaptações ao método de van der Ham (2015) foram feitas. Alguns dos passos foram reordenados, ou dispostos de forma paralela quando possível. Assim que os métodos ficaram melhor alinhados, a junção foi realizada, e o método composto final foi modelado de acordo com a Figura 4.2, sendo adotado no restante deste trabalho. Esse método pode ser interpretado como uma versão baixo nível do método de Bozkaya et al. (2009).

Serão descritos doze passos envolvidos no método. É importante esclarecer que a numeração dos passos do método (Figura 4.2) não deve ser seguida rigidamente, pois há um paralelismo no diagrama que pode ficar prejudicado.

4.1 Entender o Escopo da Análise

Neste momento devem ser realizadas uma ou mais reuniões com os *stakeholders* para entender o funcionamento do processo tal como está ("*as is*"), averiguando se já existe algum modelo que representa o fluxo como esperado.

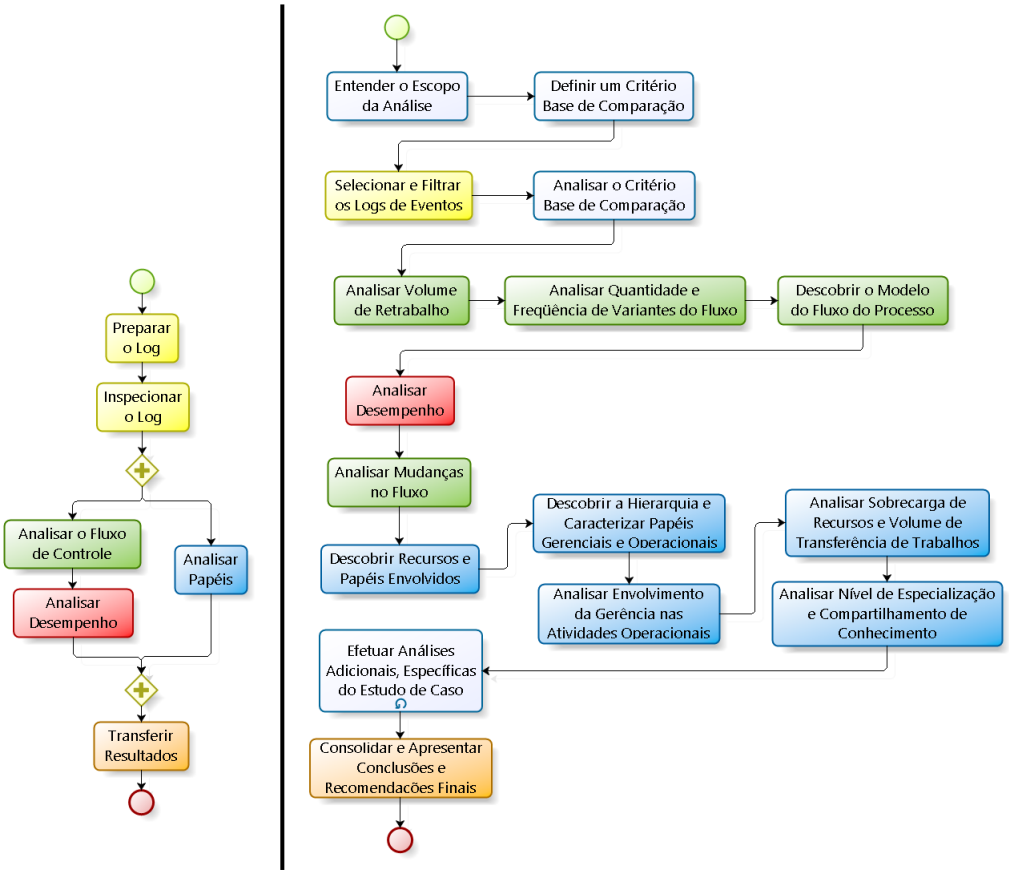

Figura 4.1: Resultado do alinhamento dos métodos de Bozkaya et al. (2009) e van der Ham (2015), onde cores iguais indicam atividades relacionadas.

Além disso, de acordo com van der Aalst (2011b), é importante iniciar os trabalhos definindo os benefícios esperados da análise que será feita. Existem basicamente três tipos diferentes de meta para um projeto de MP:

- Dirigido a dados: baseado apenas na disponibilidade dos dados. Não existe alguma pergunta a ser respondida ou algum objetivo que se espera atingir. Existe apenas a expectativa por parte dos *stakeholders* de se chegar a importantes descobertas ao analisar os dados. A característica desse tipo de projeto é investigativo e exploratório.
- Dirigido a perguntas: almeja responder perguntas específicas, como "Por que algumas requisições demoram mais para serem atendidas do que outras?", ou "Quais recursos são mais eficientes em comparação com os outros? Quais podem ser os motivos de isso ocorrer?".
- Dirigido a objetivos: focado em melhorar algum indicador em particular do processo, como reduzir custos ou melhorar o tempo de resposta.

Recomenda-se que, para organizações com menor experiência nesse tipo de projeto, os primeiros projetos sejam dirigidos a perguntas. Perguntas bem formuladas ajudam a estabelecer um foco no projeto, além de guiar os esforços de extração de dados.

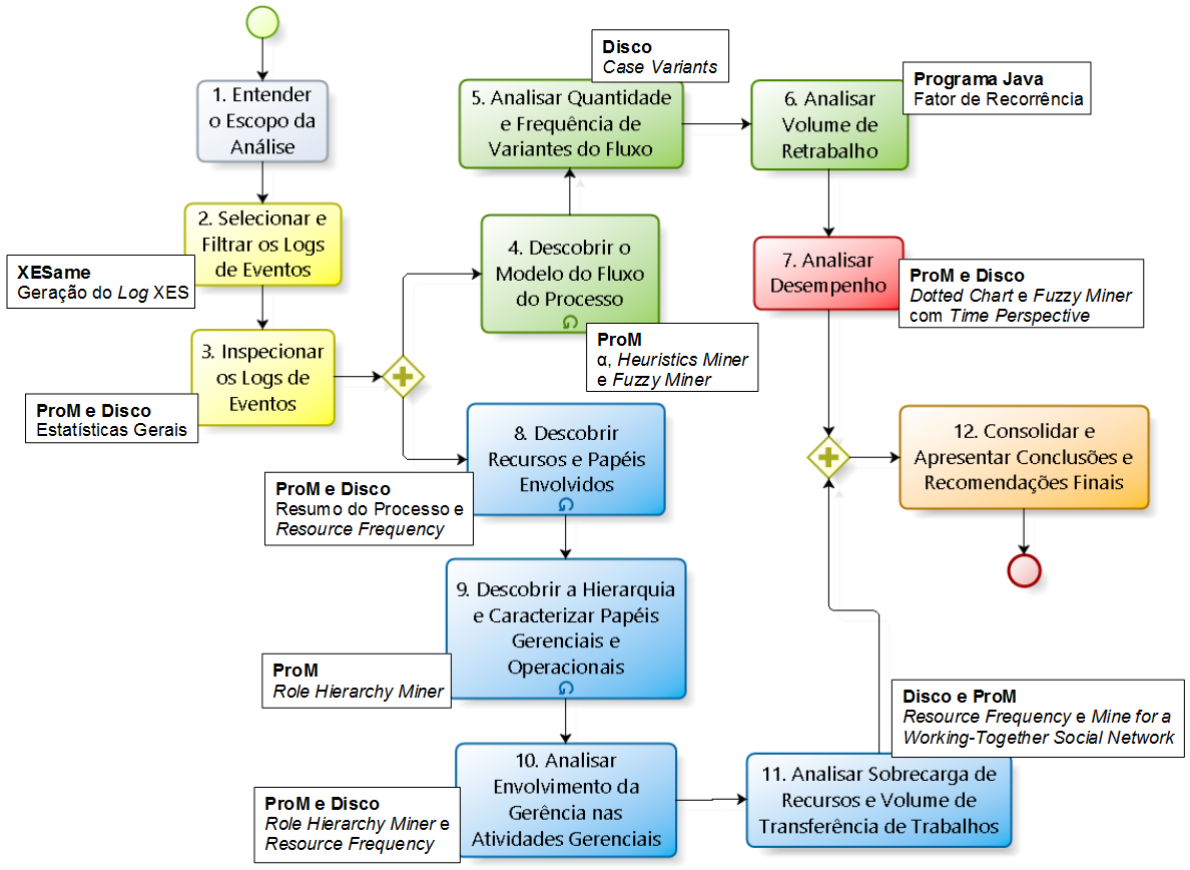

Figura 4.2: Método de MP do trabalho, incluindo ferramentas e algoritmos.

Além disso, um projeto de MP não difere de outros projetos. É necessário planejar antecipadamente os trabalhos, definir um cronograma base, planejar as fases e entregas, definir os recursos envolvidos, e acompanhar continuamente o andamento do projeto, até o seu encerramento.

4.2 Selecionar e Filtrar os *Logs*

Neste passo são selecionados todos os *logs* relevantes para fazer a extração, formatação e filtragem. A fonte de dados para extração dos *logs* normalmente será um único SI, mas em alguns casos pode ser que eles estejam fragmentados em vários sistemas, requerendo que sejam recuperados e unidos de modo que representem informações consistentes dos *traces* executados.

Nesta seção são descritos no Item 4.2.1 os formatos de entrada esperados dos dados de eventos; no Item 4.2.2 são fornecidas orientações para uma adequada identificação da instância do processo; no Item 4.2.3 os eventos de processo e seus atribudos são detalhados; no Item 4.2.4 são explicados os problemas de convergência e divergência que podem ocorrer nos dados de eventos; no Item 4.2.5 é explicada a abordagem de filtragem dos *logs*; no Item 4.2.6 são descritos os formatos de saída para geração do *log*; e no Item 4.2.7 é descrito o procedimento de geração de *log* de saída.

4.2.1 Formatos de Entrada

Os formatos de armazenamento dos logs podem variar entre sistemas. Alguns SI geram logs em arquivos texto, normalmente contendo os seguintes campos: data / hora do evento, tipo do evento (Alerta, Informativo, Depuração, Erro), número do processo que o gerou e um texto descritivo do evento. Um exemplo de *log* pode ser visto na Listagem 4.1, retratando alguns eventos de diferentes tipos relacionados com um sistema de compras. Outro exemplo de formato de log é o XML, ilustrado na Listagem 4.2 retratando os mesmos eventos do exemplo anterior. Existem ainda casos em que o *log* está armazenado num Sistema Gerenciador de Bando de Dados (SGBD), como o Oracle[1] ou MS SQL Server[2]. Nesses casos, os dados são normalmente extraídos utilizando um comando *Structured Query Language* (SQL) do tipo SELECT ou alguma rotina customizada que leia os dados da forma esperada.

Listagem 4.1: Exemplo de *log* textual de um sistema de compras

```
2016-01-01 10:12:03 INFO  7817 Criado o pedido de compra
2016-01-01 11:01:09 INFO  7825 Criado o pedido de compra
2016-01-02 10:31:42 INFO  7817 Enviado o pedido de compra
2016-01-02 10:02:10 INFO  7944 Criado o pedido de compra
2016-01-02 12:15:27 INFO  7825 Enviado o pedido de compra
2016-01-03 10:38:31 INFO  7817 Recebido o pedido de compra
2016-01-03 15:06:22 INFO  7817 Pago o pedido de compra
2016-01-03 15:38:05 DEBUG 7817 Transação encerrada com sucesso
2016-01-03 15:48:18 INFO  7825 Recebido o pedido de compra
2016-01-03 17:51:33 INFO  7825 Pago o pedido de compra
2016-01-03 17:26:03 DEBUG 7817 Transação encerrada com sucesso
2016-02-14 09:19:44 ERROR 7945 Erro ao criar pedido de compra
```

De acordo com Buijs (2010), nem todos os *logs* são gravados de uma forma descritiva, alguns SI registram apenas informações parciais dos eventos nos *logs*. Independente disso, o formato dos *logs* costuma diferir entre os sistemas. Portanto, todos os *logs* devem ser uniformizados e convertidos para um formato propício para a MP. Dentre os formatos existentes para esse intuito, dois deles tem maior destaque: MXML e XES.

[1] http://www.oracle.com/database/index.html
[2] http://www.microsoft.com/sql/default.mspx?ocid=otc-c-br-jtc–wiki

Listagem 4.2: Exemplo de *log* XML de um sistema de compras

```
<!DOCTYPE log4j:eventSet PUBLIC "-//APACHE//DTD LOG4J 1.2//EN" "log4j.
   ↪ dtd">
<log4j:eventSet version="1.2" xmlns:log4j="http://jakarta.apache.org/log
   ↪ 4j/">
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-01 10:12:03" level="INFO" pid="7817">
        <log4j:message><![CDATA[Criado o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-01 11:01:09" level="INFO" pid="7825">
        <log4j:message><![CDATA[Criado o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-02 10:31:42" level="INFO" pid="7817">
        <log4j:message><![CDATA[Enviado o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-02 12:15:27" level="INFO" pid="7825">
        <log4j:message><![CDATA[Enviado o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-03 10:38:31" level="INFO" pid="7817">
        <log4j:message><![CDATA[Recebido o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-03 15:06:22" level="INFO" pid="7817">
        <log4j:message><![CDATA[Pago o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-03 15:38:05" level="DEBUG" pid="7817">
        <log4j:message><![CDATA[Transação encerrada com sucesso]]></log4
           ↪ j:message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-03 15:48:18" level="INFO" pid="7825">
        <log4j:message><![CDATA[Recebido o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
    <log4j:event logger="com.example.Log4jXMLLayoutExample" timestamp
       ↪ ="2016-01-03 17:51:33" level="INFO" pid="7825">
        <log4j:message><![CDATA[Pago o pedido de compra]]></log4j:
           ↪ message>
    </log4j:event>
</log4j:eventSet>
```

4.2.2 Identificação da Instância do Processo

Antes de gerar um *log* dos eventos, é fundamental decidir a identificação das instâncias do processo (*case*). Quando um novo *case* é criado, uma nova instância do processo é formada. Essa instância produz um rastro de eventos ou atividades realizadas que são agrupadas e registradas no *log*. Objetos de negócio são típicos candidatos para identificação dos *cases*. Exemplos: pedidos, requisições, chamados, pacientes, notas fiscais ou até mesmo equipamentos. É importante notar que o rastro de eventos de uma instância do processo deve sempre estar relacionado com o mesmo objeto de negócio. Por exemplo, não faz sentido que, para um atendimento médico, um rastro de eventos de um *case* retrate alguns eventos na perspectiva do paciente e outros eventos na perspectiva do médico ou do departamento envolvido. Essas diferentes perspectivas seriam analisadas como uma só, gerando inconsistentes nos resultados obtidos.

A decisão da identificação dos *cases* é de grande importância para um projeto de MP, já que determina o escopo da análise realizada. Para um mesmo sistema ou processo, é possível gerar *logs* com diferentes perspectivas dependendo dessa escolha. Por exemplo, em um hospital o código do paciente pode ser definido como a identificação dos *cases*. Dessa forma, é possível analisar os tratamentos que cada paciente recebeu ao longo do tempo, ou estudar a relação dos tratamentos entre os pacientes. Se, em vez da opção anterior, o código do tratamento médico for escolhido como identificação dos *cases*, podem ser feitas análises mais específicas no fluxo de atendimento de cada tratamento, mas essa perspectiva limitará a análise numa ótica do paciente envolvido.

4.2.3 Eventos e Atributos

Uma vez definida a identificação dos *cases*, é importante determinar quais eventos serão considerados para a análise. Buijs (2010) coloca que a seleção dos eventos também influencia o foco do projeto de MP. Pode ser que apenas um subconjunto dos eventos seja relevado, para dar foco à uma parte específica do processo. Não somente isso, mas também pode ser alterado o nível de detalhamento considerado nos eventos. Um nível uniforme de detalhamento entre todos os eventos é fundamental para que se chegue a conclusões corretas e consistentes a respeito do processo analisado. A utilização de uma ontologia pode ajudar a agrupar tarefas menores numa tarefa de alto nível e, assim, garantir a uniformização do nível de detalhamento escolhido.

Os eventos de um *case* normalmente possuem os seguintes atributos:

- ***time:timestamp*:** define a sequência em que os eventos ocorrem, podendo ser armazenado nesse atributo, caso o padrão utilizado seja o XES, ou em outro campo do tipo data/hora.
- ***lifecycle:transition*:** define o passo realizado dentro do ciclo de vida do *log*. Em alguns SI, é gravada não somente a data/hora de início do evento como também a de término. Nesses casos, é possível efetuar análises mais elaboradas, considerando também o tempo efetivamente gasto na realização das tarefas. A distinção entre o evento de início (valor *"start"*) e o de término (valor *"complete"*) fica armazenado nesse atributo.
- ***org:resource*:** define o recurso responsável por realizar a tarefa.
- ***org:role*:** define a função ao qual o recurso pertence.

- ***org:group*:** define o grupo ao qual o recurso pertence.
- ***concept:name*:** define o nome da tarefa realizada.
- ***details*:** permite definir campos customizados.

4.2.4 Convergência e Divergência

Conforme mencionado no Item Identificação da Instância do Processo desta seção, cada evento refere-se a uma determinada instância do processo. No entanto, nem todos os casos práticos acontecem dessa forma.

A Figura 4.3 ilustra um exemplo do problema a ser discutido, contendo três tabelas de um SI de compras e controle de estoque. Uma das tabelas representa as entradas ao estoque relacionadas com o produto #38, cadastradas de uma forma agrupada assim como foram recebidas (representando *pallets* ou pacotes, por exemplo), juntamente com suas quantidades. Em outra tabela estão cadastradas as ordens de compra e as quantidades compradas. Cada tabela possui um campo ID que identifica cada registro de forma única. A última tabela relaciona às duas primeiras, tornando possível saber quais entradas do estoque supriram determinadas ordens de compra, ou vice-versa. É possível notar que uma ordem de compra pode não ser suprida com uma única entrada do estoque. Isso ocorre, por exemplo, com a ordem de compra 29, que é suprida pelas entradas 103, 104 e 105 do estoque. Por outro lado, nem todas as entradas do estoque são consumidas completamente na ordem de compra em que são alocadas. Um exemplo disso ocorre na entrada do estoque 106, que é alocada tanto pela ordem de compra 30 quanto pela 31.

Estoque do Produto #38

ID	Qtd.	Data / Hora
102	5	26-02-2016 14:15:28
103	15	01-03-2016 09:16:51
104	20	01-03-2016 11:23:19
105	20	02-03-2016 17:03:09
106	45	04-03-2016 14:42:27
107	52	07-03-2016 10:35:42

Ordem de Compra

ID	Qtd.	Paga em
28	5	01-03-2016 10:29:34
29	55	02-03-2016 17:49:50
30	58	05-03-2016 08:13:55
31	39	08-03-2016 09:33:17

Estoque do Produto #38 X Ordem de Compra

ID do Estoque	ID da Ordem de Compra
102	28
103	29
104	29
105	29
106	30
106	31
107	31

Figura 4.3: Modelo de tabelas de exemplo de um SI de compras e controle de estoque.

No caso de se escolher o ID da entrada do estoque como identificação da instância do processo, a Listagem 4.3 retrata como ficarão organizados os eventos.

Listagem 4.3: Organização dos eventos quando a identificação da instância do processo for o ID da entrada do estoque

```
Case EstoqueProduto [102]
- Evento 'Alocado em' OrdemCompra [28], quantidade 5, paga em 01-03-2016
   ↪  10:29:34
Case EstoqueProduto [103]
- Evento 'Alocado em' OrdemCompra [29], quantidade 55, paga em
   ↪ 02-03-2016 17:49:50
Case EstoqueProduto [104]
- Evento 'Alocado em' OrdemCompra [29], quantidade 55, paga em
   ↪ 02-03-2016 17:49:50
Case EstoqueProduto [105]
- Evento 'Alocado em' OrdemCompra [29], quantidade 55, paga em
   ↪ 02-03-2016 17:49:50
Case EstoqueProduto [106]
- Evento 'Alocado em' OrdemCompra [30], quantidade 58, paga em
   ↪ 05-03-2016 08:13:55
Case EstoqueProduto [107]
- Evento 'Alocado em' OrdemCompra [30], quantidade 58, paga em
   ↪ 05-03-2016 08:13:55
- Evento 'Alocado em' OrdemCompra [31], quantidade 39, paga em
   ↪ 08-03-2016 09:33:17
```

Note que alguns dados adicionais (recursos responsáveis pelos eventos, outros eventos envolvidos, etc.) não foram caracterizados nessa lista por questões de simplicidade. Um problema que acaba ocorrendo nessa ótica é que o evento de alocação "OrdemCompra [30]" ocorre mais de uma vez, em diferentes *cases*. Apesar de ser possível derivar essa alocação de forma fragmentada, sabendo que 45 unidades foram alocadas da entrada no estoque [106] e 13 unidades da entrada no estoque [107], em muitos casos essa dedução não é possível. Esse tipo de problema é chamado de **convergência**, sendo que a relação entre evento e processo é caracterizada como sendo do tipo **1:n**. Segue a definição de convergência de acordo com Segers (2007):

Convergência: *a mesma atividade é executada em múltiplas instâncias do processo de uma só vez.*

Esse tipo de problema afeta diretamente a análise realizada, já que o mesmo evento será contabilizado duas ou mais vezes. O desvio provocado pode acarretar uma grande disparidade nos resultados ao considerar um *log* de eventos em larga escala.

O mesmo problema pode ocorrer no sentido inverso, considerando o evento em termos da ação realizada apenas, desprezando o objeto de negócio alvo. Esse tipo de problema é chamado de **divergência**, sendo que a relação entre evento e processo é caracterizada como sendo do tipo **n:1**. Segue a definição de divergência de acordo com Segers (2007):

Divergência: *para uma instância do processo, a mesma atividade é realizada múltiplas vezes.*

Essa propriedade ocorre no *case* "EstoqueProduto [107]", já que são executados dois eventos distintos, mas da mesma tarefa de alocação de produtos no estoque. Esse tipo de situação pode gerar *loops* no modelo do fluxo minerado, que nem todos os algoritmos conseguem lidar. Alguns algoritmos de MP não lidam bem com convergência e/ou divergência. O algoritmo *Heuristics Miner* consegue lidar com ambos, mas o algoritmo α *Miner* não consegue.

4.2.5 Filtragem

Ao extrair dados dos SI envolvidos para geração do *log*, é possível aplicar alguns filtros nos dados de entrada para melhor controlar o que se deseja efetivamente analisar do processo envolvido, tanto para focar a análise em algum aspecto de maior relevância quanto para melhorar a precisão dos resultados obtidos. Buijs (2010) coloca que, basicamente, essa filtragem pode ser feita em termos de *cases* ou de eventos. Além dessas duas opções, podem ser aplicados filtros mais específicos para eliminar *cases* que contenham um determinado evento, ou *cases*/eventos que possuam alguma característica em particular.

Uma questão importante relacionada com a filtragem de eventos é a janela de tempo considerada. Ao extrair dados de um sistema para análise, deve ser definido o período de seleção desses dados históricos. A Figura 4.4 ilustra uma sequência de cinco atendimentos ao longo de um período de cinco meses de um suposto suporte técnico. O processo envolvido é estruturado, e o atendimento segue uma sequência previsível e constante. Se imaginarmos que os dados considerados para geração do *log* compreendem o intervalo de meses entre Fevereiro e Abril, teríamos um corte similar ao representado pelas duas linhas vermelhas. É possível notar que os chamados 2, 3 e 4 estariam completamente inseridos nesse período, mas os chamados 1 e 5 não.

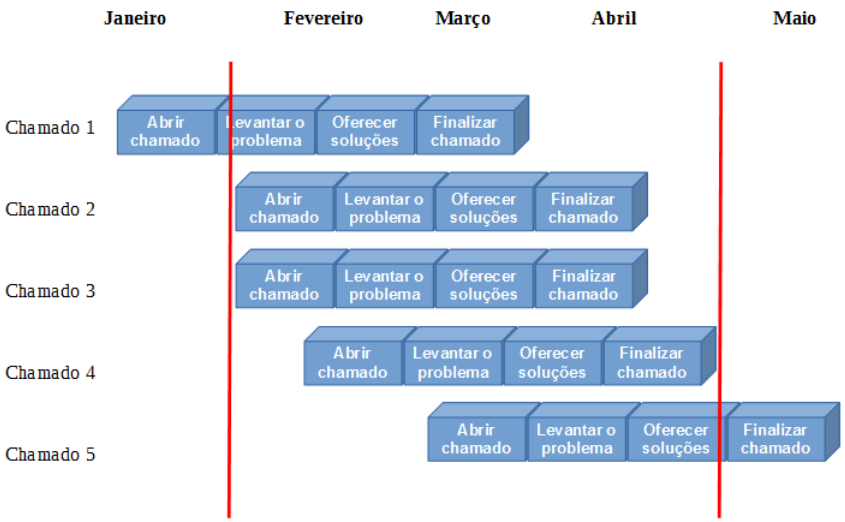

Figura 4.4: Ilustração de uma sequência de atendimentos técnicos ao longo do período de Janeiro a Maio.

Ao minerar o *log* correspondente a esse período de três meses no ProM utilizando o algoritmo α *Miner*, o modelo gerado está representado na metade de cima da Figura

4.5. Note a diferença de minerar utilizando o mesmo algoritmo, mas considerando os cinco meses de dados, ilustrado na metade de baixo da mesma figura. Como existem *traces* incompletos no primeiro caso, foram gerados arcos adicionais no modelo de cima, sugerindo desvios no fluxo que, de fato, não existem na prática.

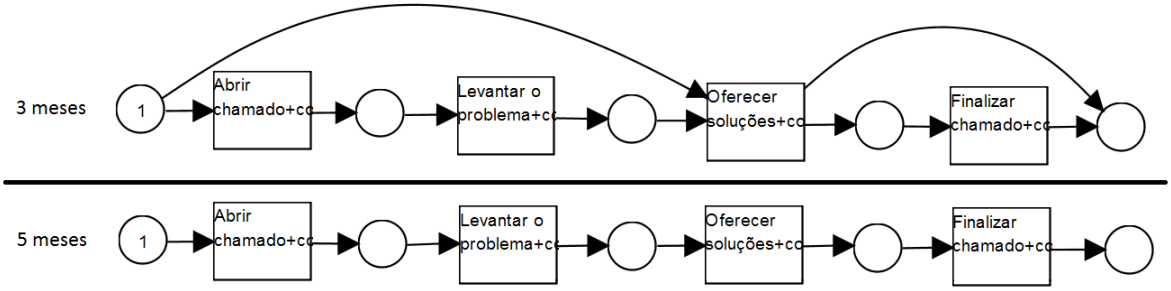

Figura 4.5: Rede de Petri minerado de um exemplo de suporte técnico para o período de três e cinco meses.

Esse tipo de preocupação é muito comum nos projetos de MP. Uma abordagem muito utilizada para estudar essa escolha é levantar o percentual de *traces* incompletos na janela de tempo definida. Caso o percentual seja relativamente pequeno, é possível que a distorção seja pequena e, portanto, desprezível. Caso seja significativo, deve-se utilizar algum mecanismo para reduzir o percentual levantado. Podem, por exemplo, ser utilizados filtros adicionais ao do tempo da janela para reduzir ao máximo os *traces* incompletos, ou pode ser desenvolvida uma rotina customizada que extraia os dados de um período apenas para os *traces* completos. De qualquer forma, essa é uma questão relevante ao considerar os filtros aplicados na geração do *log* final.

Existem também casos em que se deseja efetuar uma análise minuciosa nos eventos do fluxo, mas apenas para alguns *traces* que tenham características peculiares. Nesse caso, serão utilizados filtros para selecionar uma amostra pequena dos dados de entrada, mas que tenham relativa significância para o objetivo em questão. A busca pelos *traces* adequados para esse tipo de análise pode ser demorada, mas extremamente necessária para atingir um nível de precisão adequado.

4.2.6 Formatos de Saída

O formato MXML, implementado na linguagem XML, foi criado em 2005 por Van Dongen e Van der Aalst com o objetivo de servir de padrão para análise de eventos na MP (Buijs, 2010). Apesar desse formato ter servido de valioso modelo para a análise de processos, alguns problemas foram detectados ao longo do tempo. Um deles, por exemplo, está relacionado com a semântica de atributos adicionais armazenados nos eventos do *log*, que podem ser definidos de uma forma excessivamente flexível e livre, tornando-os um tanto arbitrários e de difícil padronização. Por causa desse e outros problemas, um novo formato de nome XES foi posteriormente criado, implementado na linguagem XML de uma forma significativamente diferente do seu antecessor. De acordo com Buijs (2010), a revisão 3 da versão 1.0 desse novo formato foi atualizada pela última vez em 2009, sendo

que atualmente está disponível na versão 2.0[3]. O novo formato XES foi concebido para resolver as lacunas do MXML e, assim, se tornar o padrão efetivo no registro e análise na MP. A Listagem 4.4 ilustra o conteúdo parcial de um arquivo XES XML.

Listagem 4.4: Exemplo de parte de um *log* XES com um *trace* e um evento

```
<trace>
  <string key="concept:name" value="Pedido1"/>
  <float key="pedido:valorTotal" value="2142.38"/>
  <event>
    <string key="concept:name" value="Criar"/>
    <string key="lifecycle:transition" value="complete"/>
    <string key="org:resource" value="Marcos"/>
    <date key="time:timestamp" value="2016-01-03T15:30:00.000+01:00"/>
    <float key="pedido:valorAtual" value="2142.38"/>
    <string key="details" value="DetalhesPedidoCompra">
      <string key="requestedBy" value="Paula"/>
      <string key="supplier" value="Fluxi Inc."/>
      <date key="expectedDelivery" value="2016-01-12T
         ↪ 12:00:00.000+01:00"/>
    </string>
  </event>
</trace>
```

4.2.7 Geração do *Log* de Saída

Após definir a identificação dos *cases*, definir o critério de seleção dos eventos e mapear os atributos que serão preenchidos em cada evento, o *log* de saída já pode ser gerado, preferencialmente no formato XES. Caso os dados a serem extraídos estiverem armazenados num SGBD[4], uma ferramenta recomendada para gerar o arquivo de *log* é o XESame, mencionado na Seção 2.1.6. O uso dessa ferramenta será ilustrado a seguir. A Figura 4.6 retrata uma visão parcial do Modelo Entidade Relacionamento (MER) das tabelas definidas no SGBD MySQL de um suposto sistema. A tabela **LogAuditoria** contém os dados históricos de eventos do sistema, e a tabela **Usuario** contém os usuários cadastrados para utilizá-lo, sendo que existe uma relação entre as duas tabelas do tipo **n:1**.

Primeiramente foi baixado e instalado o pacote do ProM versão 6.5.1, da qual o XESame faz parte. Após executar o XESame, na tela inicial, aba Ferramentas*Connection*, é possível configurar os parâmetros de conexão ao banco de dados. Esses valores variam de ambiente para ambiente, dependendo do servidor e porta em que o MySQL escuta, caminho do driver *Java Database Connectivity* (JDBC), usuário e senha de acesso ao banco de dados, entre outros. O botão *Test Connection* pode ser utilizado para averiguar se os parâmetros de conexão estão corretos.

[3]http://www.win.tue.nl/ieeetfpm/lib/exe/fetch.php?media=shared:downloads:xes-r12d2.pdf

[4]Alguns *drivers Open Database Connectivity* (ODBC) e *Java Database Connectivity* (JDBC) conseguem ler até mesmo arquivos CSV ou arquivos texto como se fossem tabelas de um banco de dados relacional

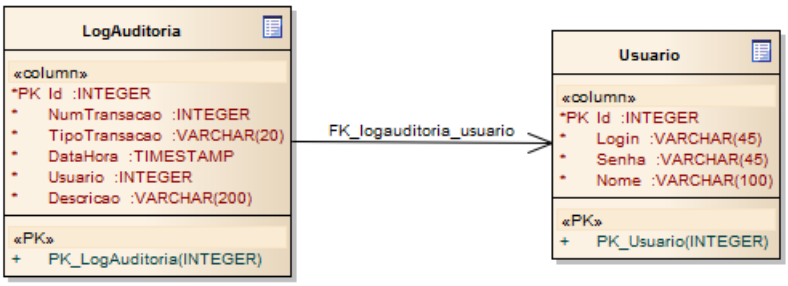

Figura 4.6: Modelo de tabelas de exemplo relacionado com o *log* de um processo.

Na aba Mapeamento*Definition* são definidos os parâmetros para mapear os dados do SGBD em dados do *log* XES. Para este exemplo em particular, as seguintes configurações foram feitas:

- Item ***Log***: atributo *concept:name* ficou com o valor 'Exemplo' e a propriedade *From* com o valor LogAuditoria.
- Item ***Trace***: atributo *concept:name* ficou com o valor NumTransacao, a propriedade *From* com o valor LogAuditoria e a propriedade *TraceID* ficou com o valor NumTransacao.
- Item ***Event***: este item foi criado manualmente abaixo do item ***Trace*** com o nome *Event*. Além disso, o atributo *concept:name* ficou com o valor Descricao, o atributo *lifecycle:transition* com o valor *'complete'*, o atributo *org:resource* com o valor Usuario.Login, o atributo *time:timestamp* com o valor DataHora, a propriedade *From* com o valor LogAuditoria, a propriedade *TraceID* com o valor NumTransacao, a propriedade *EventOrder* com o valor DataHora [{yyyy-MM-dd hh:mm:ss}] e um novo *Link* foi adicionado manualmente contendo o valor Usuario ON Usuario.Id = LogAuditoria.Usuario.

Na aba Execução*Settings* são definidos os parâmetros para efetivamente gerar o *log* XES, que, no contexto desta ferramenta, recebe a denominação de Conversão, já que estão sendo convertidos dados em um formato para outro. Ao clicar no botão *Execute Conversion* será aberta uma janela que apresentará o resultado da execução. Essa saída poderá ser posteriormente consultada na aba Execução*Console*. As Figuras A.1, A.2, A.3 e A.4 do Apêndice A ilustram esses passos e as configurações realizadas.

4.3 Inspecionar os *Logs* de Eventos

O objetivo deste passo é obter um entendimento preliminar e relativamente superficial do processo a partir dos *logs* selecionados no passo anterior. Para que isso seja possível, serão mineradas e analisadas algumas estatísticas gerais, como número de *cases* e papéis, totais de tarefas e eventos, máximo / mínimo / média de eventos por *case*, entre outras. Além disso, serão efetuados alguns estudos iniciais a respeito do processo, como identificar as rotas mais comuns e as especiais (menos frequentes) no fluxo. Neste passo também podem ser aplicados filtros adicionais para refinar o conjunto de *logs* selecionados, permi-

tindo assim agilizar o entendimento do processo ao focar em pequenos subconjuntos de eventos, ou naqueles que tenham alguma característica específica em comum.

Esse primeiro nível de análise pode ser feito tanto pela ferramenta ProM quanto pela Disco. Por exemplo, as Figuras 4.7, 4.8 e 4.9 apresentam parte dessas informações. Na Figura 4.7, acionada na aba *Log Visualizer\Dashboard*, é possível ver, entre outros, quantos(as) processos, *cases*, eventos, classes de tarefas, tipos de eventos (normalmente *complete* e, possivelmente, *start*) e classes de tarefas que iniciaram o fluxo existem no *log* minerado. Quando as classes de tarefas que efetivamente finalizam o fluxo forem conhecidas, é possível diferenciar quantos *cases* ainda estão em aberto, já que poderão ser contabilizadas pelas ocorrências dessas classes de tarefas em comparação com as outras, que não finalizaram efetivamente o fluxo. Na Figura 4.8, acionada na aba *Log Visualizer\Inspector\Explorer*, é possível ver cada *case* superficialmente, através de uma notação de cores para cada classe de tarefa. Já na Figura 4.9, acionada na aba *Log Visualizer\Summary*, é possível ver, entre outros, quais são as classes de tarefas, identificar as que iniciaram e as que finalizaram o fluxo, o percentual em que elas ocorrem, quais são os recursos envolvidos, identificar os recursos que iniciaram e os que finalizaram o fluxo. Esses valores estão representados em duas colunas: *Occurrences (absolute)*, contendo a quantidade absoluta de ocorrências, e *Occurrences (relative)*, contendo o percentual que o valor absoluto representa no total geral de ocorrências.

Figura 4.7: Aba *Log Visualizer\Dashboard* do ProM.

Outras informações podem também ser visualizadas por essas ferramentas (ProM e Disco), como as diferentes variantes do fluxo, *cases* ativos ao longo do tempo e duração total dos *cases*. Todas essas informações são úteis para ter uma visão geral do processo, orientar as análises seguintes e avaliar a necessidade de aplicar algum filtro adicional com a intenção de evitar análises tendenciosas ou desvirtuadas de algum modo.

Figura 4.8: Aba *Log Visualizer**Inspector**Explorer* do ProM.

4.4 Descobrir o Modelo do Fluxo do Processo

Neste passo são selecionados o(s) algoritmo(s) de descoberta do modelo, executado(s) para minerar o *log* e gerado(s) o(s) modelo(s) do fluxo do processo. É importante que o modelo gerado tenha uma qualidade mínima esperada, tanto para servir de valioso insumo para o entendimento do processo quanto para apoiar adequadamente as análises seguintes. O modelo do fluxo tem influência direta nos passos seguintes da análise, já que muitas delas o requerem como insumo para efetuar cálculos e também agregar-lhe novas informações. Por isso, o modelo é considerado a espinha dorsal da análise de processos.

Para que seja gerado um bom modelo, uma das primeiras preocupações é considerar as características do processo, como existência de repetições, concorrências decorrentes de escolhas, incompletude e ruído. Alguns algoritmos não lidam adequadamente com essas propriedades, e a escolha de um ou mais algoritmos mais adequados depende dessas limitações, entre outros.

A utilização de diferentes algoritmos pode ser útil para comparar os resultados e validar/revisar parâmetros de configuração dos algoritmos. É possível que alguns algoritmos gerem o modelo em apenas uma determinada notação, por exemplo Rede de Petri, e outros gerem em BPMN. Para evitar um esforço constante de tradução das notações, os modelos podem ser convertidos para uma notação comum, quando possível.

A respeito da notação de representação, é importante lembrar que algumas limitações dos algoritmos em termos de tratamento de casos especiais, como repetições, concorrência decorrente de escolha, *loops* aninhados, *splits* e *joins* desbalanceados e sincronização parcial, tem relação direta com a notação utilizada. De acordo com van der Aalst (2016), "descoberta de processo é, por definição, limitada pelo poder de expressividade da linguagem alvo, ou seja, o viés representacional". Aparentemente, a escolha de um viés representacional mais adequado pode beneficiar a abordagem utilizada, mas, principalmente nos casos especiais, pode também diminuir a expressividade da notação, introduzindo

várias atividades duplicadas ou silenciosas (atividades que não podem ser observadas). Portanto, a utilização de um viés representacional mais adequado não necessariamente elimina essa limitação.

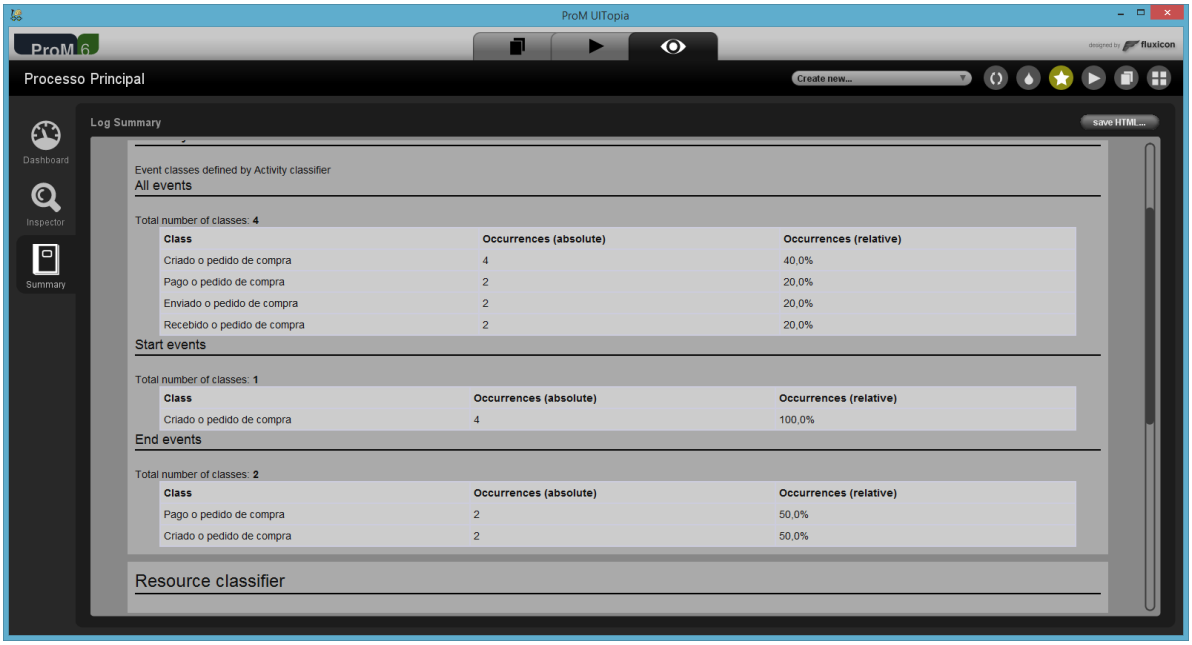

Figura 4.9: Aba *Log Visualizer\Summary* do ProM.

A partir da experiência adquirida nos trabalhos de experimentação empírica, são descritas na Tabela 4.1 algumas indicações de aplicabilidade dos algoritmos α *Miner*, *Fuzzy Miner* e *Heuristics Miner* (Seção 2.1.5).

Tabela 4.1: Algoritmos e indicações de aplicabilidade.

Algoritmo	**Indicações de Aplicabilidade**
α *Miner*	- Não consegue lidar com ruído (dados incorretos ou incompletos) nos *logs* e processos que tenham tarefas repetidas ou *loops* - Pode servir de referência inicial para as análises posteriores
Fuzzy Miner	- Indicado para processos mais complexos, já que consegue abstrair detalhes do fluxo - Consegue lidar com ruído nos *logs*, processos que contenham tarefas repetidas e *loops*
Heuristics Miner	- Consegue lidar com ruído nos *logs* e processos que tenham *loops*, mas não consegue lidar com tarefas repetidas

Após a escolha dos algoritmos, o *log* deve ser minerado para gerar o modelo correspondente. Uma vez gerado o modelo, deve ser feita uma avaliação para averiguar se a qualidade do mesmo está satisfatória. É importante lembrar que não existe um único

modelo, correto, que descreve o processo. De acordo com van der Aalst (2016), existem muitos modelos ou visões da mesma realidade, e o processo sendo analisado pode ainda passar por mudanças ao longo do tempo.

4.4.1 Quatro Critérios Concorrentes de Qualidade

De acordo com van der Aalst (2016), definir critérios de qualidade de um modelo de processo é uma tarefa difícil e caracterizada por muitas dimensões. Em particular, quatro dimensões são consideradas: *fitness*, simplicidade, precisão e generalização. Uma visão alto nível dessas dimensões está ilustrada na Figura 4.10.

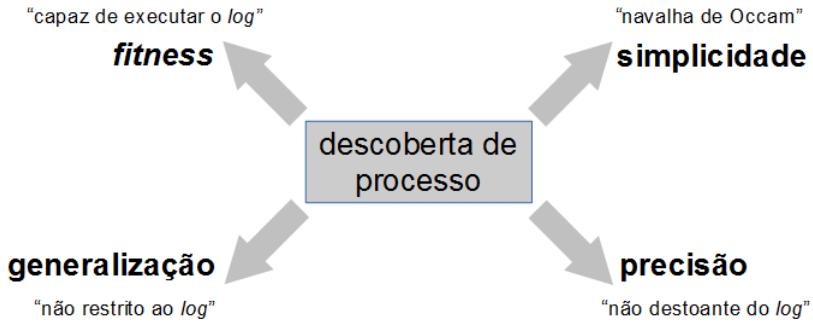

Figura 4.10: Quatro dimensões que ajudam a definir a qualidade de um modelo.

Um modelo que tenha a característica de *fitness* significa que representa o comportamento existente no *log* correspondente, ou seja, ao executar um *log* de acordo com um modelo já existente, 100% de *fitness* indica que todos os *traces* estão contemplados pelo modelo. Existem diferentes formas de pontuar o nível de *fitness* de um modelo, dependentes basicamente de decisões de projeto. Ao executar um *log* de acordo com o modelo, questões como o peso da penalidade ao ter que pular uma atividade, ou de não conseguir concluir o fluxo do modelo, variam de caso a caso. Mas são decisões necessárias para poder avaliar o nível de *fitness* de um modelo gerado. De acordo com van der Aalst (2016), o maior interesse está no *modelo 80/20*, ou seja, no modelo que descreve 80% do comportamento do *log* em termos de *fitness*, mas se mantém simples ao desconsiderar os outros 20% (variações menos relevantes do fluxo ou ruído do *log*).

Outra medida a ser considerada é a simplicidade do modelo, que se refere à "Navalha de Occam", um princípio atribuído a William de Ockham que dita: "não se deve aumentar, além do necessário, o número de entidades necessárias para explicar qualquer coisa" (van der Aalst, 2016). No contexto de descoberta do modelo, quanto mais simples for o modelo que represente o comportamento existente no *log* correspondente, melhor será. Existem algumas métricas que podem ser utilizadas para pontuar a complexidade do modelo. Uma delas é o número de nós e arcos do grafo subjacente. Mas existem outras métricas mais complexas que também podem ser utilizadas, como a "entropia"do modelo.

Apesar dessas duas medidas serem necessárias, não são suficientes para determinar a qualidade de um modelo. Um exemplo disso pode ser visto na Figura 4.11, em que é ilustrada uma rede Petri no formato "flor". Esse modelo, apesar de atender a um *log* em termos de *fitness* e simplicidade, é extremamente abrangente, além de não oferecer nenhuma informação útil do processo além da lista de atividades envolvidas.

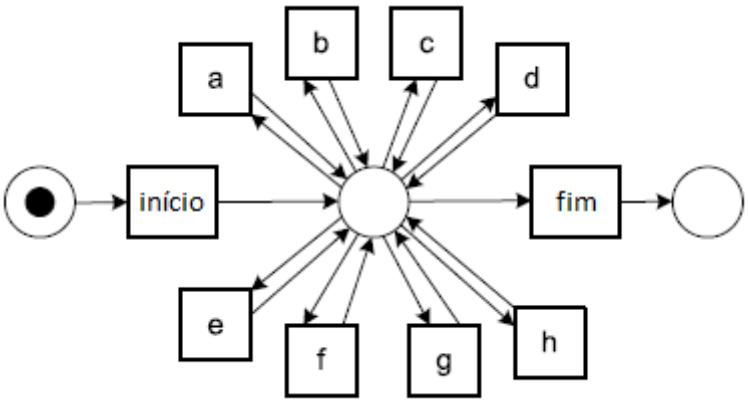

Figura 4.11: Rede de Petri de exemplo no formato “flor” (van der Aalst, 2016).

No outro extremo está o modelo do tipo "enumeração", ilustrado na Figura 4.12. Esse tipo de modelo tem no início uma grande disjunção exclusiva e, ao final, uma grande junção exclusiva, simplesmente listando todas as sequências possíveis, cada uma num fragmento separado para representar os *traces* do *log*.

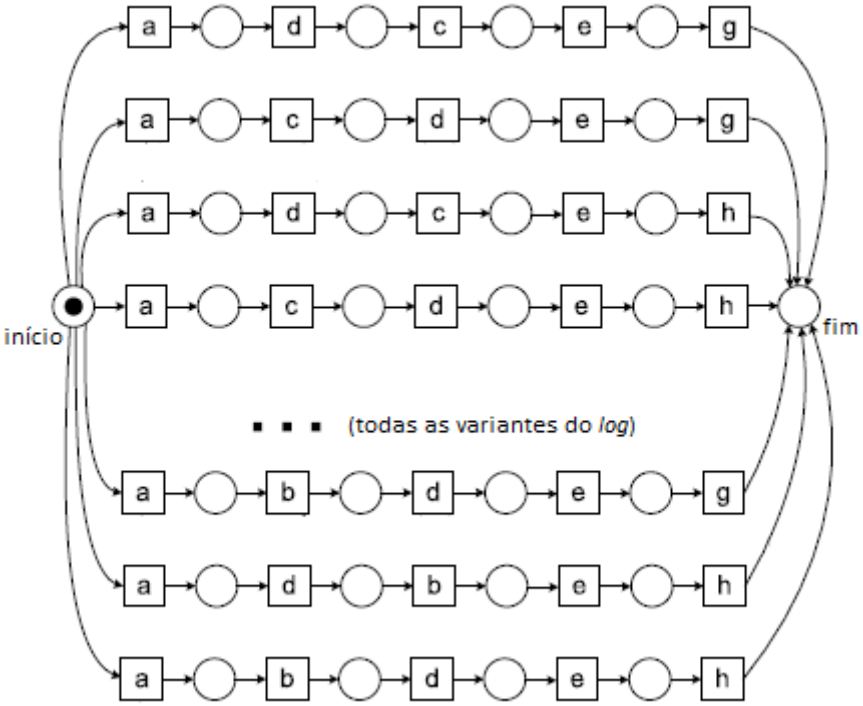

Figura 4.12: Rede de Petri de exemplo no formato “enumeração” (van der Aalst, 2016).

Esses dois extremos de modelos mostram a necessidade de duas dimensões adicionais, que são: precisão e generalização. Um modelo é preciso se ele não permite comportamento "demais". No exemplo do modelo do tipo "enumeração", fica nítido que é preciso, apesar de excessivamente complexo. Um modelo é generalizado quando contempla não apenas a pequena amostra da qual foi minerado, mas que possui flexibilidade suficiente pra contem-

plar outras variantes similares. O modelo do tipo "flor" é nitidamente genérico, apesar de ser pouco expressivo. Para alcançar uma qualidade esperada, o modelo minerado deve atingir um equilíbrio entre essas duas medidas, ou seja, entre precisão e generalização.

Portanto, ao descobrir o modelo do processo a partir da mineração do *log*, é importante validar a qualidade obtida através de uma verificação posterior. Uma das formas de fazer isso é executar um teste de conformidade entre o *log* e o modelo gerado, e assim quantificar o valor *fitness* entre 0 e 1. Além disso, existem técnicas que permitem metrificar a simplicidade do modelo e o equilíbrio entre precisão e generalização. Uma métrica mais simples consiste na quantidade de nós e arcos existentes no modelo gerado. Outras métricas mais sofisticadas podem levar em conta o nível de organização ou entropia do modelo. Portanto, existem diferentes medidas para avaliar essas quatro dimensões de qualidade, que podem depender dos objetivos do analista e estarem limitadas pelo viés representacional.

4.5 Analisar Quantidade e Frequência de Variantes do Fluxo

Neste passo é levantada a quantidade de variantes que o fluxo possui e a frequência dessas variantes. Uma variante representa um caminho único de sequência de atividades, do início ao fim do processo. Isso significa que, por exemplo, diferenças no número de execuções de *loops* do processo determina diferentes variantes. Essa análise de variantes permite identificar o(s) caminho(s) crítico(s) e a importância das outras variantes no fluxo.

A ferramenta Disco possui uma forma amigável de apresentar as variantes dos *cases*. Ao clicar na aba *Cases* é possível ver uma lista das variantes e o percentual em que cada variante ocorre. Ao clicar em uma delas, é também possível ver o fluxo que a variante representa, conforme Figura 4.13.

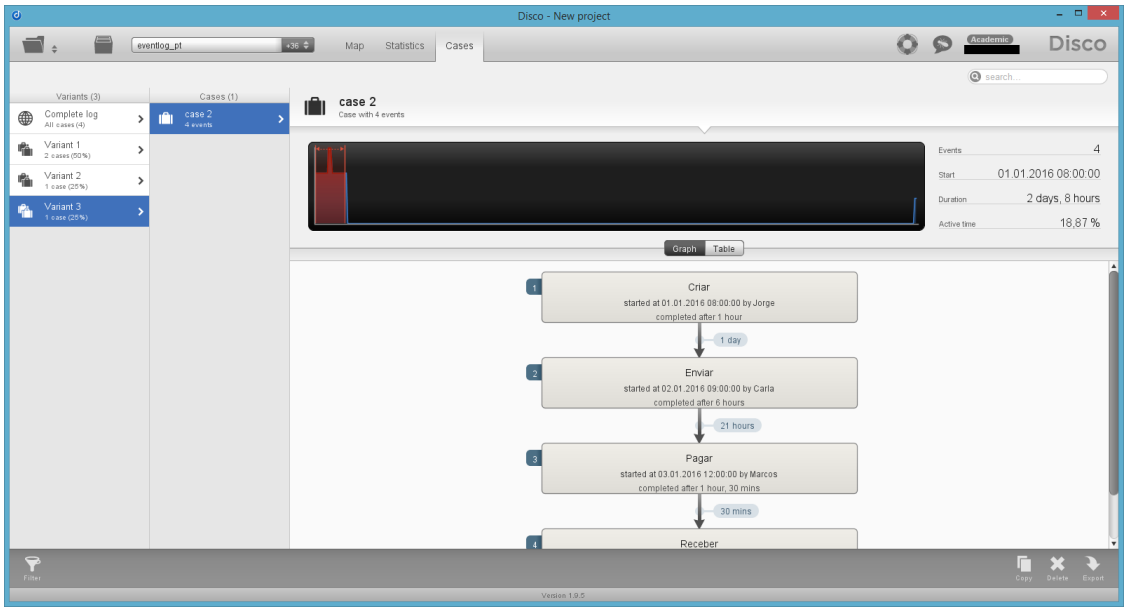

Figura 4.13: Aba *Cases**Variant 2* do Disco.

4.6 Analisar Volume de Retrabalho

O conceito de retrabalho pode ser expresso como o processo de retificação de erros na realização de tarefas de modo que alguns requisitos sejam atendidos. No contexto de processos, essa situação pode ser identificada, de forma simplificada, através da análise estatística de repetições das mesmas tarefas nos *cases* de um processo. Portanto, neste passo é identificada a frequência com que essa recorrência ocorre.

O cálculo de recorrência das tarefas não deve ser feito apenas dividindo o número de execuções de cada tarefa pelo número de *cases*, já que nem sempre a tarefa ocorre em todos os *cases*. Por exemplo, se a tarefa 'Redistribuir o incidente' de um atendimento técnico ocorre em apenas um *case* do processo, mas numa quantidade de repetições relativamente alta, o cálculo citado pode não caracterizar o problema se o número de *cases* em que a tarefa não ocorre for significativo.

Portanto, uma abordagem mais recomendada é dividir o número de instâncias de cada tarefa pelo número de *cases* em que ela efetivamente ocorre. Existem diferentes formas de efetuar esse cálculo, dependendo da ferramenta ou da disponibilidade e formato dos *logs* do SI. Para este trabalho, foi desenvolvida uma rotina customizada na linguagem de programação Java que realiza essa contabilização, já que não foi encontrada funcionalidade similar no ProM ou Disco. A rotina desenvolvida pode ser visualizada na listagem do Apêndice B.

O programa apenas lista as tarefas que tenham fator de recorrência maior que 1, já que esses casos sugerem que houve algum nível de retrabalho. Um exemplo de saída desse programa pode ser visto na Listagem 4.5.

Listagem 4.5: Saída da rotina que contabiliza o fator de recorrência para um *log* de exemplo

```
Parsing XES file 'exemplo.xes' ...

Cases: 4

Recurrence Factor
-----------------

Recebido o pedido de compra:      1,5 (taskCount=3/
   ↪ casesWithTask=2) [MAX: caseId=127, count=2]
Enviado o pedido de compra:       2 (taskCount=4/casesWithTask
   ↪ =2) [MAX: caseId=127, count=3]
```

É possível notar que a tarefa "Enviado o pedido de compra" ocorreu num fator de recorrência de valor 2, ou seja, para 2 *cases* ("*casesWithTask*") em que ela ocorreu (e não 4, que é o total de *cases* desse *log*), foram realizadas 4 vezes ("*taskCount*") a mesma tarefa, sugerindo a existência de retrabalho. Além disso, essa tarefa teve um máximo de 3 ocorrências ("*count*") num único *case*, identificado pelo ID "127" ("*caseId*").

4.7 Analisar Desempenho

Neste passo são levantados alguns indicadores de desempenho do processo, como tempo médio / mínimo / máximo de conclusão de um *case*, caminho no fluxo em que mais se investiu tempo, atividade com maior tempo gasto, tempo de realização de tarefas versus tempo de espera entre tarefas.

Por exemplo, para identificar a tarefa com maior tempo gasto, pode ser utilizada a ferramenta Disco, aba *Map**Performance*, com a opção *Show* definida para *Total Duration*, conforme ilustrado na Figura 4.14. Através dessa figura, é possível notar que a tarefa com maior tempo gasto foi a "Enviar", com um tempo total investido de 14,2 horas de trabalho.

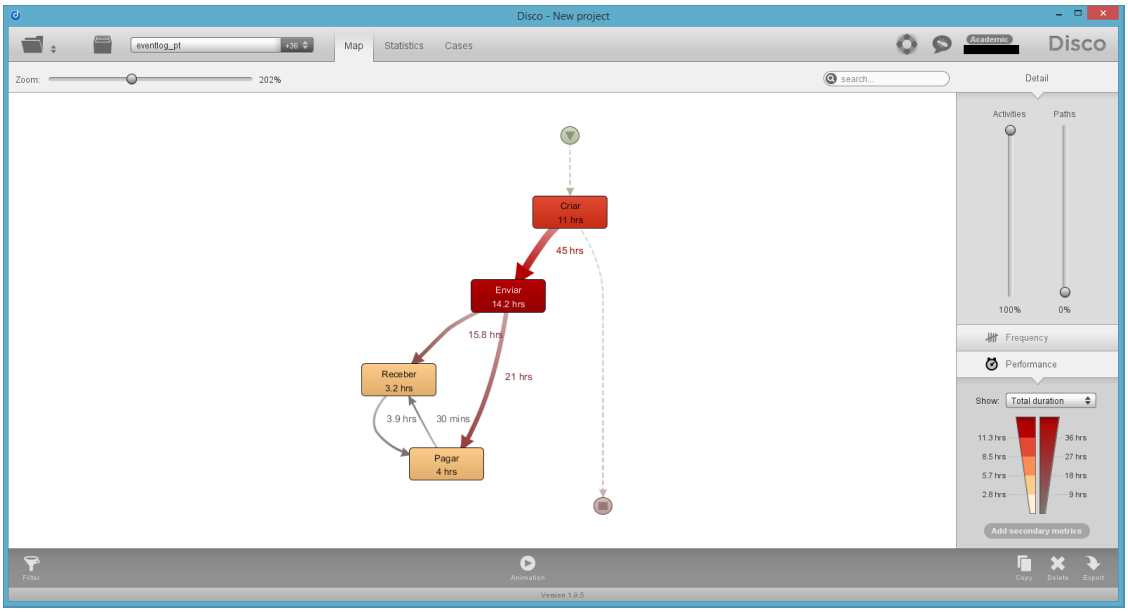

Figura 4.14: Aba *Map**Performance* do Disco.

4.8 Descobrir Recursos e Papéis Envolvidos

Neste passo são descobertos os recursos e papéis que participam do processo. No ProM, essa informação fica acessível na aba *Log Visualizer**Summary*, e no Disco, fica na aba *Statistics**Resource*.

4.9 Descobrir a Hierarquia e Caracterizar Papéis Gerenciais e Operacionais

Neste passo são diferenciados os recursos ou papéis gerenciais dos operacionais dentro da estrutura organizacional. Em alguns casos, o próprio nome do recurso sugere essa diferenciação (e.g., Gerente e Técnico). Uma forma de caracterizar essa diferenciação é através do tipo de tarefa que o recurso desempenha. Estes são alguns exemplos de tarefas tipicamente gerenciais: "Designar o chamado", indicando que ele decide para quem devem

ser encaminhados os chamados; "Negociar o orçamento", indicando que ele negocia junto ao cliente o orçamento estimado; "Acompanhar a entrega", indicando que ele acompanha o andamento dos trabalhos; "Avaliar desempenho", indicando que ele faz uma avaliação de desempenho do time envolvido.

Quando o recurso executa alguma atividade administrativa ou gerencial ao longo do processo, existe um forte indício de que o papel que ele exerce na organização, ou no processo, seja efetivamente gerencial. Para realizar essa análise, um algoritmo recomendado é o *Role Hierarchy Miner*, disponível no ProM, já que o gráfico de saída gerado pelo algoritmo permite identificar quais recursos executam cada tarefa.

4.10 Analisar Envolvimento da Gerência nas Atividades Gerenciais e Operacionais

Neste passo é avaliado se os recursos gerenciais atuam exclusivamente nas atividades de administração, coordenação e acompanhamento do processo, entre outras, ou se também participam nas tarefas operacionais. Para que esta análise seja possível, é necessário que o especialista em mineração tenha conhecimento prévio do perfil das tarefas realizadas no processo, ou seja, se elas são de caráter gerencial, operacional, ou ambos. Para realizar essa análise, é necessário quantificar a frequência com que os recursos ou papéis gerenciais executam as atividades operacionais, em comparação com todas as atividades realizadas.

4.11 Analisar Sobrecarga de Recursos e Volume de Transferência de Trabalhos

Neste passo é analisado se um ou mais recursos em particular estão sobrecarregados em comparação aos demais. Para quantificar essa sobrecarga, pode ser utilizada a ferramenta Disco, aba *Statistics\Resource*, que quantifica o nível de participação do recurso no processo, conforme ilustrado na Figura 4.15. É possível notar, nesse exemplo em particular, que quase todos os recursos estão sendo utilizados de forma equilibrada.

Se a quantidade de recursos envolvidos for relativamente grande, pode ser que esse gráfico perca legibilidade. Para contornar esse problema, pode ser gerado o mesmo tipo de gráfico a partir do atributo *org:role* ou *org:group*, caso estejam preenchidos. Dessa forma, os dados representados estarão resumidos de acordo com esse agrupamento, conforme ilustrado na Figura 4.16. Nesse exemplo, as áreas de "Suprimentos" e "Planejamento" são mais atuantes, apesar de não representarem uma diferença significativa em relação às outras áreas.

Além disso, deve ser analisado o volume de transferência de trabalhos entre recursos. Para realizar essa análise, um algoritmo recomendado é o *Working-Together Social Network*, disponível no ProM, já que o gráfico de saída gerado pelo algoritmo permite identificar o nível de repasse direto (de uma atividade para a seguinte) quanto indireto (de uma atividade para alguma atividade seguinte, mesmo que existam atividades entre elas) de trabalho entre os pares de recursos.

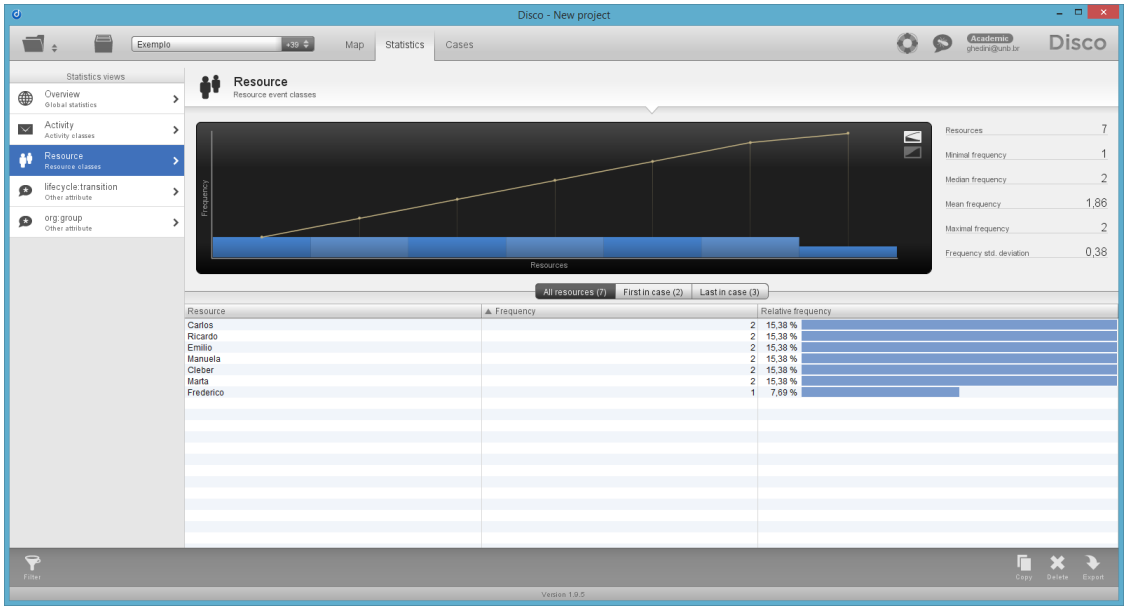

Figura 4.15: Aba *Statistics**Resource* do Disco.

4.12 Consolidar e Apresentar Conclusões e Recomendações Finais

Neste passo são apresentados e revisados os resultados com o cliente, procurando diferenciar comportamento esperado de comportamento não esperado, e confirmando as análises feitas a respeito do processo analisado. Finalmente, deve ser feita uma apresentação com as descobertas realizadas, conclusões e recomendações do diagnóstico, com o objetivo de prover um entendimento comum dos resultados e permitir que o cliente revise o seu processo e os SI envolvidos.

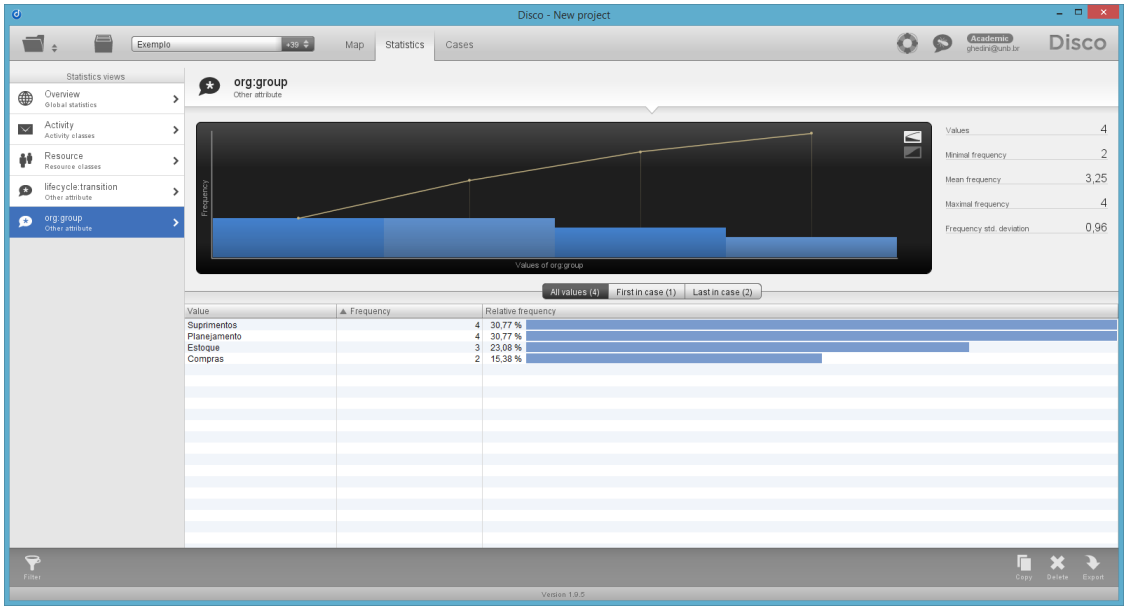

Figura 4.16: Aba *Statistics**Group* do Disco.

Neste capítulo foi apresentado o método de MP proposto através do modelo BPMN, uma descrição dos passos envolvidos, ilustrações e exemplos relacionados. No próximo capítulo, será apresentada uma aplicação prática deste método num caso real de um Tribunal de Justiça, para entender de forma mais clara como o método pode ser efetivamente utilizado num projeto de MP.

Capítulo 5

Cenário de Aplicação e Resultados

Neste capítulo será apresentado um cenário de aplicação do método proposto no Capítulo 4, seguindo os passos da Figura 4.2. No final deste capítulo, serão apresentados os resultados e respondidas as perguntas que motivaram esse projeto.

A aplicação do método envolveu a análise de um sistema de informação que foi desenvolvido para a gestão de pessoas em empresas públicas. O sistema consiste de uma solução de Gestão Tradicional de Pessoas e Gestão Estratégica, que permite gerenciar as informações tratadas pela área de gestão de pessoas e relacionadas, contemplando desde a admissão do servidor até a sua exoneração ou aposentadoria. O sistema foi analisado a partir de dados reais de um Tribunal Federal. O processo escolhido para análise foi o controle de férias, já que envolve todos os servidores do órgão e existem importantes perguntas a serem respondidas. Na sequência serão detalhados os passos do método proposto.

5.1 Entender o Escopo da Análise

Foram feitas algumas reuniões de levantamento com técnicos envolvidos no controle de férias, para entender como o processo de férias é realizado, quais papéis e passos estavam envolvidos, entender quais problemas prejudicam ou dificultam a correta execução do processo, entre outros.

Os seguintes papéis foram citados nesse levantamento: servidor, gestor e analista de RH. O servidor é a parte interessada em tirar férias, marcando-a conforme sua necessidade. O gestor é o responsável por homologar as férias e, quando necessário, revisar a marcação feita pelo servidor. E o analista de RH faz as concessões de férias e auxilia em todo o processo, acompanhando e revisando todos os passos para garantir que as férias sejam marcadas e gozadas dentro dos limites e regras previstos na legislação aplicável.

Como ainda não existia um diagrama do processo de férias modelado na organização, o fluxo foi desenhado a partir dos relatos dos técnicos que participaram dessas reuniões. O modelo resultante pode ser visto na Figura 5.1.

O sistema utilizado para o controle de férias não faz o controle do início de cada atividade, apenas registra o momento em que elas são finalizadas. No entanto, os *stakeholders* envolvidos também desejavam realizar análise da duração das atividades para atender a alguns critérios de auditoria. Portanto, para que essa dimensão não fosse completamente

perdida, foi solicitado que fossem atribuídas durações médias a todas as atividades, de modo que esses valores sejam também considerados nas análises de desempenho do processo. É importante ressaltar que essa medida pode gerar considerável diferença entre as medições de desempenho encontradas e as reais.

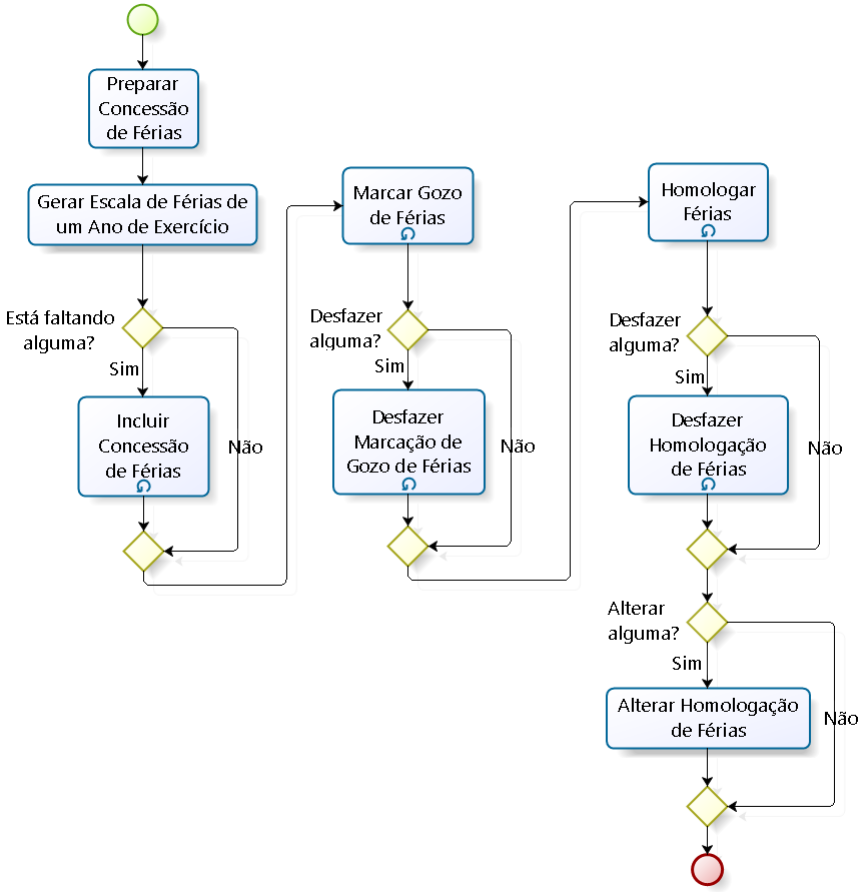

Figura 5.1: Processo de férias modelado utilizando BPMN.

5.2 Selecionar e Filtrar os *Logs*

A seleção e filtragem dos *logs* envolveu um processo empírico de análise do sistema envolvido. Uma consultora do sistema prestou esclarecimentos sobre como o sistema funcionava, em que local os dados estavam armazenados, quais eram os dados históricos mantidos e em qual formato. De acordo com esse repasse, descobriu-se que o sistema gera um *log* de auditoria de todas as mudanças efetuadas nos dados do sistema, sejam elas de inclusão, alteração ou exclusão, conforme ilustrado na Figura 5.2.

A partir dessas informações, e considerando que o sistema armazena os dados no banco de dados pós-relacional Caché[1], os dados foram avaliados para saber se estavam prontos para a extração. O conjunto completo das tabelas relacionadas com os dados históricos de

[1] http://www.intersystems.com/cache

eventos está ilustrado na Figura 5.3. Como era necessário fazer alguns cálculos relativamente complexos sobre esses dados, eles foram transformados e armazenados em uma nova tabela que estava mais adequada para a extração. Essa transformação foi feita através de uma rotina customizada e desenvolvida na linguagem nativa do banco de dados, de nome *Caché Object Script*. A tabela final utilizada para armazenar os dados transformados, de nome "Ferias.Log", está ilustrada na Figura 5.4.

Assunto	ObjId	Acao	DataHora	Usuario	Usuario Portal
Concessão Férias	27231	create	2013-10-16 08:42:42	NULL	NULL
Homologação de Férias	48293	create	2013-12-13 12:44:49	157	NULL
Homologação de Férias	48293	edit	2013-12-16 11:16:06	157	NULL
Concessão Férias	27568	create	2013-10-16 08:45:58	NULL	NULL
Gozo de Férias	18566	create	2013-10-30 10:25:07	NULL	2224
Homologação de Férias	46491	create	2013-11-05 17:22:35	NULL	68
Homologação de Férias	46492	edit	2014-07-10 10:20:24	277	NULL

Figura 5.2: Dados de *logs* de auditoria.

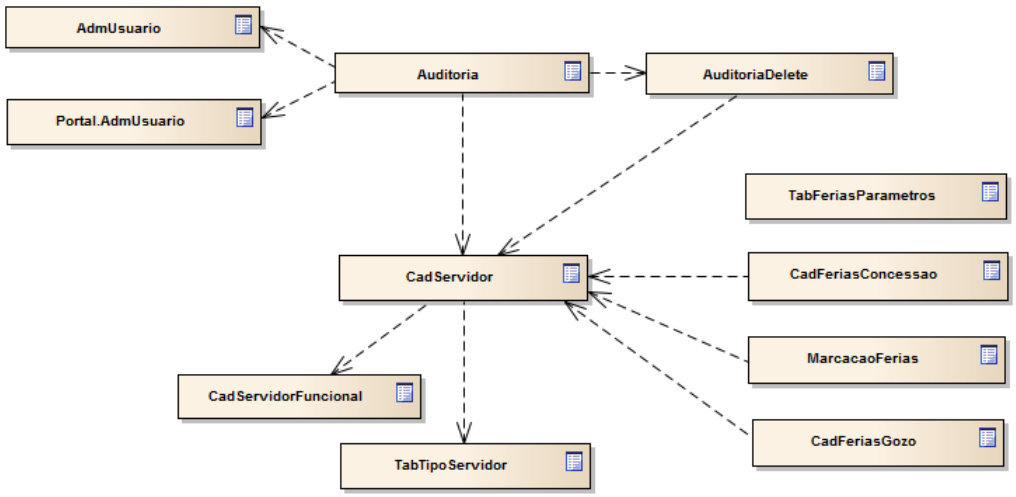

Figura 5.3: Modelo das tabelas envolvidas na transformação dos dados históricos de eventos.

Um aspecto importante na transformação desses dados foi a escolha da informação que identificaria a instância do processo. Considerando o fluxo de férias desenhado, o que se mostrou mais adequado para o caso em questão foi considerar a junção dos campos "ano de exercício" e "número de identificação do servidor", armazenados na coluna "*CaseId*". Ou seja, uma instância do processo representa todos os passos envolvidos para conceder,

marcar, homologar e ajustar as férias de um servidor em um determinado ano (2014-2015). Após esse tratamento dos dados, foi desenvolvido um comando SQL que extraísse os dados relevantes da tabela final. Esse comando pode ser visualizado na Listagem 5.1.

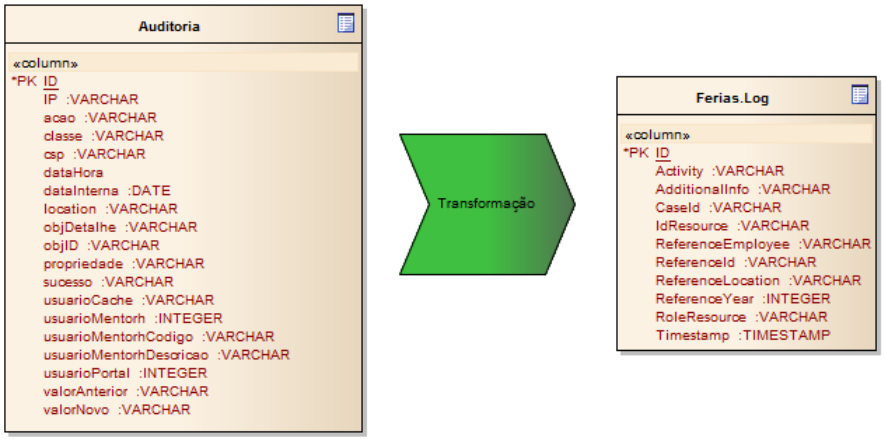

Figura 5.4: Tabela Auditoria transformada para Ferias.Log.

Listagem 5.1: Comando SQL para extração dos dados relevantes

```
  SELECT CaseId,
         Activity,
         Timestamp,
         RoleResource,
         IdResource,
         ReferenceId,
         ReferenceYear,
         ReferenceLocation,
         ReferenceEmployee,
         AdditionalInfo
    FROM Ferias.Log
   WHERE (ReferenceYear >= 2014)
     AND (ReferenceYear <= 2015)
ORDER BY ReferenceYear, ReferenceEmployee, Timestamp
```

Por uma questão de evolução do sistema, notou-se que, apenas nos últimos anos de exercício, os dados estão armazenados integralmente no banco de dados. Portanto, os *logs* foram filtrados para considerar apenas os anos de exercício de 2014 e 2015. O comando SQL contem algumas colunas básicas necessárias para a análise de *logs* de processos: identificação da instância do processo, nome da atividade, data/hora e recurso envolvido.

Através do comando SQL da Listagem 5.1, foi utilizada a ferramenta XESame para gerar o arquivo XES correspondente, de acordo com a explicação do Item 4.2.7 relativo ao uso do XESame. Após gerado o arquivo XES, esse foi aberto no Disco para uma validação inicial.

Como já mencionado, os dados de histórico não possuem duração de tempo, apenas o evento que registra o término de cada atividade. Ao abrir o arquivo XES no Disco, foi possível visualizar o nome de cada tarefa envolvida no processo. A partir dessa lista de nomes, foi revisada a definição das durações médias definidas pelos *stakeholders* para garantir que todas as tarefas foram consideradas, e criada a Tabela 5.1. Como a tarefa "Preparar Concessão de Férias" é realizada, na prática, de uma só vez para todos os servidores, esse trabalho centralizado foi diluído pra cada servidor com um tempo proporcional de 1 minuto. E como a tarefa "Conceder Férias" é realizada de uma só vez, em lote, para todos os servidores, também foi definida com um tempo proporcional de 1 minuto por servidor. A tarefa "Alterar Homologação de Férias" foi definida com um tempo maior porque não é feita pelo próprio servidor, por uma questão de limitação do sistema, e sim apenas pela equipe de RH.

Tabela 5.1: Tempo estimado das tarefas.

Tarefa	**Tempo Estimado**
Preparar Concessão de Férias	1 min
Conceder Férias	1 min
Marcar Gozo de Férias	20 min
Desfazer Marcação de Gozo de Férias	20 min
Homologar Férias	30 min
Desfazer Homologação de Férias	30 min
Alterar Homologação de Férias	60 min

Para que fosse gerado um *log* de eventos contendo essas durações, a montagem do arquivo XES não pôde mais ser feita através do XESame. Portanto, primeiramente foi feita nova extração dos dados através do comando SQL da Listagem 5.1, utilizando um cliente de banco de dados que conseguisse exportar dados consultados no formato CSV. A Figura 5.5 apresenta um trecho desse arquivo gerado.

Em seguida, foi desenvolvida uma rotina customizada na linguagem de programação Java que inserisse um evento de fechamento para cada atividade, respeitando a tabela de durações definida anteriormente (vide Apêndice C). Através dessa rotina, o arquivo XES resultante foi gerado.

Esse arquivo foi novamente aberto no Disco para uma análise inicial e também para avaliar a necessidade de efetuar alguma filtragem adicional nos dados de origem. Um dos aspectos considerados nessa avaliação foi a incompletude dos dados de eventos. Mas a filtragem dos *logs* para a janela de tempo determinada não gerou *traces* incompletos, pois,

os dados já possuíam como identificador o ano e servidor ao qual pertencem, ou seja, os dados coletados de cada ano já representavam os *traces* completos desse período.

```
CaseId;Activity;Timestamp;RoleResource;IdResource;ReferenceId;ReferenceY
2014/0000000001;Homologar Férias;2014-08-25 10:44:24;RH;RHPE3874;8491712
2014/0000000001;Homologar Férias;2014-08-25 10:45:06;RH;RHPE3874;8491717
2014/0000000001;Homologar Férias;2014-12-12 11:11:20;RH;RHPE3874;8770539
2014/0000001002;Preparar Concessão de Férias;2013-10-16 07:45:58;RH;RH;7
2014/0000001002;Conceder Férias;2013-10-16 08:45:58;RH;RH;7663308;2014;3
```

Figura 5.5: Trecho do arquivo CSV gerado a partir da extração do banco de dados da tabela Ferias.Log.

5.3 Inspecionar os *Logs* de Eventos

Os *logs* foram verificados usando dados estatísticos para obter um melhor entendimento do processo que eles representam. Na Tabela 5.2 estão representados alguns totais gerais do processo, como total de *cases* ou instâncias do processo, total de eventos e total de tarefas. De acordo com esses valores, existem, em média, 12,27 eventos (32.504/2.649) por *case*, e como esse valor supera a quantidade de tarefas, ficou subentendido que algumas delas ocorrem mais de uma vez nos *cases*.

Tabela 5.2: Estatísticas gerais.

Informação	**Valor**
Quantidade de *Cases*	2.649
Quantidade de Eventos	32.504
Quantidade de Tarefas	7
Quantidade de Recursos Envolvidos	1.533

A Tabela 5.3 retrata algumas quantidades por *case*. É possível notar uma grande variação na quantidade de eventos por *case*, num intervalo entre 2 e 90. Além disso, existem alguns *cases* que só passaram por duas tarefas, indicando que não foram concluídos e esses servidores provavelmente não tiraram férias.

Na Tabela 5.4 é possível identificar as tarefas mais recorrentes nos *cases*. É possível notar um grande investimento de esforço na marcação do gozo e homologação de férias, com frequências de eventos muito próximas. Além disso, a alteração da homologação de férias também exigiu um investimento considerável de esforço.

Na Tabela 5.5 é possível visualizar a quantidade de recursos por papel exercido no processo. Em média existe 1 gestor para cada 6,6 servidores (1.285/195), e 1 analista de RH para cada 24,2 servidores (1.285/53) envolvidos no processo.

Tabela 5.3: Estatísticas por *case*.

Informação	**Tipo**	**Valor**
Qtd. de Eventos por *Case*	Mínimo	2
	Médio	12,27
	Máximo	90
Qtd. de Tarefas por *Case*	Mínimo	2
	Médio	5
	Máximo	7

Adicionalmente, foi realizada análise das atividades que iniciaram ou finalizaram os *cases*. Apenas uma atividade iniciou todos os *cases*: "Preparar Concessão de Férias". E as seguintes atividades encerraram a maioria dos *cases*: "Homologar Férias" (51,71%), "Alterar Homologação de Férias" (22,46%) e "Desfazer Marcação de Gozo de Férias" (19,37%).

5.4 Descobrir o Modelo do Fluxo do Processo

5.4.1 Algoritmo α *Miner*

Foi então feita a mineração do *log* na ferramenta ProM para descobrir o fluxo do processo. De acordo com as estatísticas iniciais, a maioria das atividades ocorrem em quase todos os *cases*, com exceção da atividade "Desfazer Homologação de Férias". Portanto, pelo menos essas atividades principais deveriam estar presentes no fluxo descoberto. Além disso, como existe uma grande variação na quantidade de eventos por *case*, foi feita uma análise para saber se alguns *cases* deveriam ser filtrados para evitar distorções. No entanto, as variantes de menor quantidade de eventos são bem frequentes, representando significativa importância para a análise do processo. E as variantes de maior quantidade de eventos, apesar de ocorrerem pontualmente, apenas representam fluxos similares aos da maioria dos *cases*, mas com mais ou menos repetições dos *loops* existentes. Portanto, ambos os extremos foram mantidos.

Para efeito de análise, foi primeiramente utilizado o algoritmo α *Miner* no ProM, mas, como existem atividades repetidas, o resultado obtido não foi satisfatório, conforme a Figura 5.6. O modelo gerado, justamente pela limitação desse algoritmo neste caso em

particular, aparenta estar incompleto ou inconsistente, já que não existe conexão entre o início e fim do fluxo, entre outros problemas.

Tabela 5.4: Totais de ocorrências por tarefa.

Tarefa	**Total de Ocorrências**
Preparar Concessão de Férias	2.649
Conceder Férias	2.866
Marcar Gozo de Férias	8.095
Desfazer Marcação de Gozo de Férias	3.797
Homologar Férias	8.607
Desfazer Homologação de Férias	176
Alterar Homologação de Férias	6.314

Tabela 5.5: Totais de recursos por papel envolvido.

Papel Envolvido	**Total de Recursos**
Servidores	1.285
Gestores	195
Analistas de RH	53

5.4.2 Algoritmo *Heuristics Miner*

Em seguida, foi utilizado o algoritmo *Heuristics Miner*. O primeiro teste foi de minerar o *log* utilizando os valores padrões, que servem de bom ponto de partida para a maioria dos casos. Para a versão 6.5.1 do ProM, esses valores são: *Relative-to-best*=5, *Dependency*=90, *Length-one-loops*=90, *Length-two-loops*=90, *All tasks connected*=*yes*, *Long distance dependency*=*no*, *Ignore loop dependency thresholds*=*yes*. O nível de *fitness* alcançado foi muito baixo, um valor de 47,6%. Portanto, a opção *All tasks connected* foi

desmarcada (ficando com o valor *no*), para evitar que algumas relações sejam estabelecidas apenas pela melhor opção possível, apesar de diminuirem o nível de conformidade calculado pelo algoritmo. O modelo gerado pode ser visto na Figura 5.7. Apesar do nível de *fitness* melhorar para 65,4%, o modelo possui várias tarefas desconexas do fluxo, tornando-o pouco satisfatório.

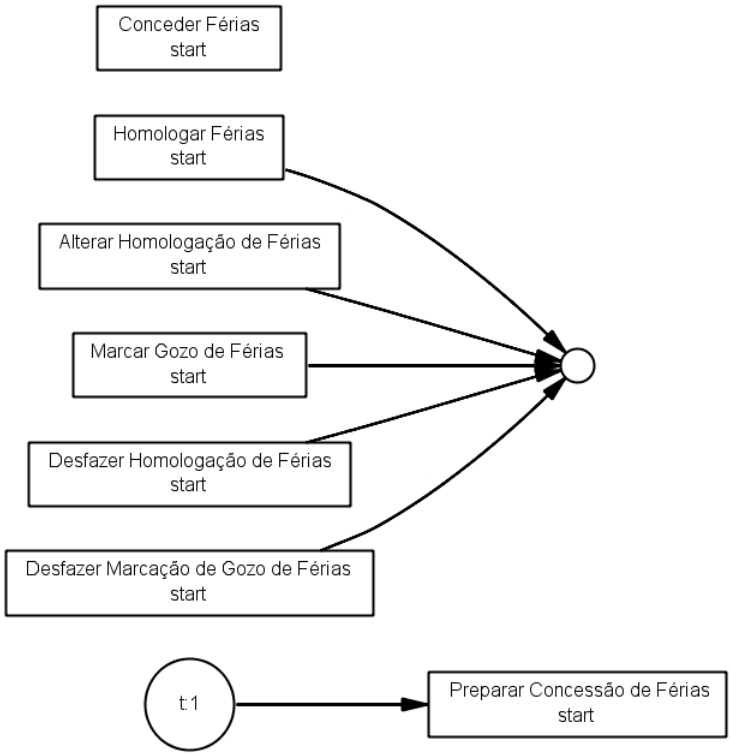

Figura 5.6: Modelo do processo de férias minerado com o algoritmo α *Miner*.

Uma das razões do *fitness* ser baixo em ambos os casos ocorre porque, assim como no algoritmo α *Miner*, o *Heuristics Miner* espera que o fluxo tenha uma única tarefa de início e uma única de fim. Neste caso, existe uma tarefa de início, a "Preparar Concessão de Férias", mas existem várias de término, já que as vezes o processo de férias envolve alterações ou remarcações no período inicialmente definido, mas esses desvios não são uma premissa do fluxo. Ao minerar o *log* utilizando o mesmo algoritmo, mas na versão 5.2 do ProM, a janela de saída apresenta as mensagens "*The END task is possible not unique?*" e "*Use the 'Add an artificial END-task' option*". Portanto, no ProM 6.5.1, o *log* foi submetido à opção "*Add End Artificial Events*", e foi adicionado um evento ao final de todos os *traces* de nome "*ArtificialEnd*".

Posteriormente, o *log* foi submetido ao mesmo algoritmo com as opções padrões. O nível de *fitness* alcançado foi maior, 64,7%, mas ainda insatisfatório. Portanto, novamente a opção *All tasks connected* foi desmarcada, e o novo fluxo gerado pode ser visto na Figura 5.8. Neste caso, o nível de *fitness* atingiu o valor de 70,9%, o que já pode ser considerado satisfatório. Ou seja, não atingiu a proporção 80/20, mas é aceitável num contexto de controle de férias. No entanto, uma tarefa ainda está desconexa do fluxo, a "Desfazer Homologação de Férias".

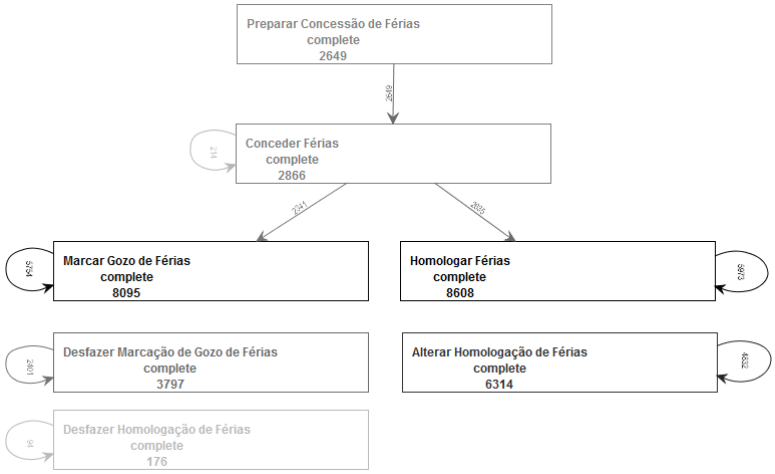

Figura 5.7: Modelo do processo de férias minerado com o algoritmo *Heuristics Miner*: *All tasks connected=no*.

Esses resultados estão relacionados com a forma como o algoritmo calcula as relações entre as tarefas, conforme descrição da Seção 2.1.5. A Tabela 5.6 apresenta todas as frequências dessa métrica para o gráfico gerado na Figura 5.8.

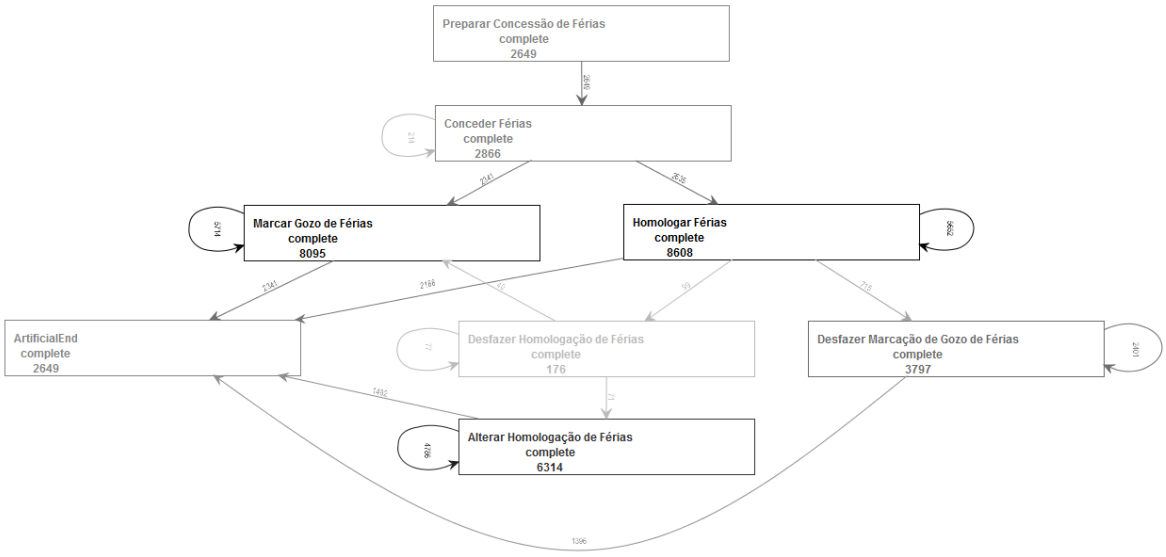

Figura 5.8: Modelo do processo de férias minerado com o algoritmo *Heuristics Miner*: Com *ArtifialEnd* e *All tasks connected=yes*

A tabela de dependências computada está representada na Tabela 5.7. A tarefa de início do fluxo, "Preparar Concessão de Férias", foi marcada com verde escuro na tabela, e a tarefa de término, "*Artificial End*", com vermelho escuro. As células de maior valor da linha foram marcadas com verde claro, e as células contendo valores positivos na linha que atendam aos *thresholds* definidos foram marcadas em azul. As marcações de verde claro e azul correspondem aos arcos existentes no modelo da Figura 5.8, com exceção dos arcos que definem *loops* para a própria tarefa.

Tabela 5.6: Métrica de frequência de dependência entre as tarefas.

$\|a >_w b\|$	Preparar Concessão de Férias	Conceder Férias	Marcar Gozo de Férias	Desfazer Marcação de Gozo de Férias	Homologar Férias	Desfazer Homologação de Férias	Alterar Homologação de Férias	Artificial End
Preparar Concessão de Férias	0	2.649	0	0	0	0	0	0
Conceder Férias	0	214	2.296	0	339	0	1	16
Marcar Gozo de Férias	0	0	4.191	1.092	2.467	1	193	151
Desfazer Marcação de Gozo de Férias	0	0	1.333	1.349	68	56	478	513
Homologar Férias	0	1	105	893	3.709	77	2.452	1.379
Desfazer Homologação de Férias	0	0	22	52	48	18	32	4
Alterar Homologação de Férias	0	2	148	411	1.976	24	3.158	595
Artificial End	0	0	0	0	0	0	0	0

No entanto, é possível perceber pelos resultados anteriores que, apesar do nível de *fitness* ter aumentado, o diagrama ainda não representa da melhor forma o fluxo correspondente. Por exemplo, a tarefa "Alterar Homologação de Férias" ocorre várias vezes após a tarefa "Homologação de Férias", mas elas não possuem uma ligação direta. Uma forma de atingir um modelo que conecte essas duas tarefas é de relaxar o parâmetro *Length-two-loops*. A Figura 5.9 ilustra como fica o modelo minerado, se esse parâmetro estiver definido com o valor 0, mas o nível de *fitness* também cai muito, ficando em 11,5%. Além disso, o modelo perdeu legibilidade, já que praticamente todas as conexões entre as tarefas estão representadas, apesar de algumas ocorrerem numa frequência relativamente baixa.

Portanto, o parâmetro *Relative-to-best* foi aumentado para 10 e o valor *Dependency* foi diminuído para o valor 50, relaxando os critérios de conexão entre as tarefas. O modelo final gerado pode ser visualizado na Figura 5.10. Esse modelo atingiu o maior nível de *fitness*, 71,1%, dentre todos os valores testados nesse algoritmo.

Apesar dessas adaptações e parametrizações terem sido feitas, o modelo gerado ainda não ficou satisfatório, já que, entre outros, não conectou as duas tarefas citadas anteriormente, ou seja, apesar do algoritmo *Heuristics Miner* ser bastante útil para tratar, entre outros, ruídos ou casos excepcionais, neste caso em particular, na qual existe uma frequência considerável de *loops* entre pares de atividades, com valores bem próximos de ligação nos dois sentidos, acaba se tornando uma atividade dispendiosa encontrar os melhores valores de *thresholds* para sintonizar apropriadamente o algoritmo e alcançar um bom resultado.

Tabela 5.7: Tabela de dependência entre as tarefas.

$\Rightarrow_W$	Preparar Concessão de Férias	Conceder Férias	Marcar Gozo de Férias	Desfazer Marcação de Gozo de Férias	Homologar Férias	Desfazer Homologação de Férias	Alterar Homologação de Férias	Artificial End
Preparar Concessão de Férias	0,0000	0,9996	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Conceder Férias	-0,9996	0,0000	0,9996	0,0000	0,9912	0,0000	-0,2500	0,9412
Marcar Gozo de Férias	0,0000	-0,9996	0,0000	-0,0993	0,9180	-0,8750	0,1316	0,9934
Desfazer Marcação de Gozo de Férias	0,0000	0,0000	0,0993	0,0000	-0,8576	0,0367	0,0753	0,9981
Homologar Férias	0,0000	-0,9912	-0,9180	0,8576	0,0000	0,2302	0,1075	0,9993
Desfazer Homologação de Férias	0,0000	0,0000	0,8750	-0,0367	-0,2302	0,0000	0,1404	0,8000
Alterar Homologação de Férias	0,0000	0,2500	-0,1316	-0,0753	-0,1075	-0,1404	0,0000	0,9983
Artificial End	0,0000	-0,9412	-0,9934	-0,9981	-0,9993	-0,8000	-0,9983	0,0000

5.4.3 Algoritmo *Fuzzy Miner*

Por fim, foi utilizado o algoritmo *Fuzzy Miner*, já que o algoritmo consegue lidar com atividades repetidas, além de permitir controlar o que é considerado relevante e deverá ser mantido no modelo minerado. O resultado obtido pode ser visualizado na Figura 5.11.

O nível de *fitness* fornecido pelo ProM ao executar o algoritmo de mineração é de 95,54%, ou seja, ele consegue executar 95,54% do *log* de eventos do arquivo XES. Esse nível de *fitness* pode ser considerado satisfatório, inclusive ao comparar com os resultados anteriores. O modelo gerado retrata todas as tarefas envolvidas e a quantidade de arestas que as conectam estão em número relativamente pequeno, fazendo com que o modelo seja simples e legível. A precisão e generalização do modelo estão bem balanceadas, já que a maioria dos eventos do *log* foram atendidos com pouca especificidade na representação dos diferentes fluxos dos *cases*, ao mesmo tempo em que especifica de forma consistente os diferentes fluxos possíveis.

5.5 Analisar Quantidade e Frequência de Variantes do Fluxo

Para um melhor entendimento do fluxo minerado, foi feita uma análise da quantidade de variantes e total de ocorrências das mesmas. A Figura 5.12 ilustra as variantes do fluxo mineradas pela ferramenta Disco. As três primeiras variantes representam aproximada-

mente 30% dos *cases*, e praticamente todas as restantes representam individualmente menos que 1% dos *cases*.

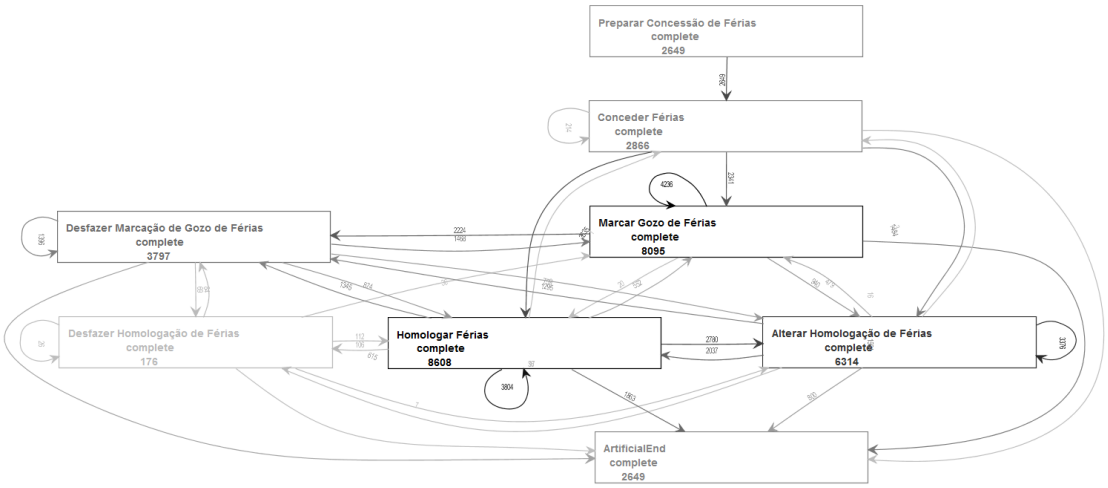

Figura 5.9: Modelo minerado com o algoritmo *Heuristics Miner* e com o parâmetro *Length-two-loops*=0.

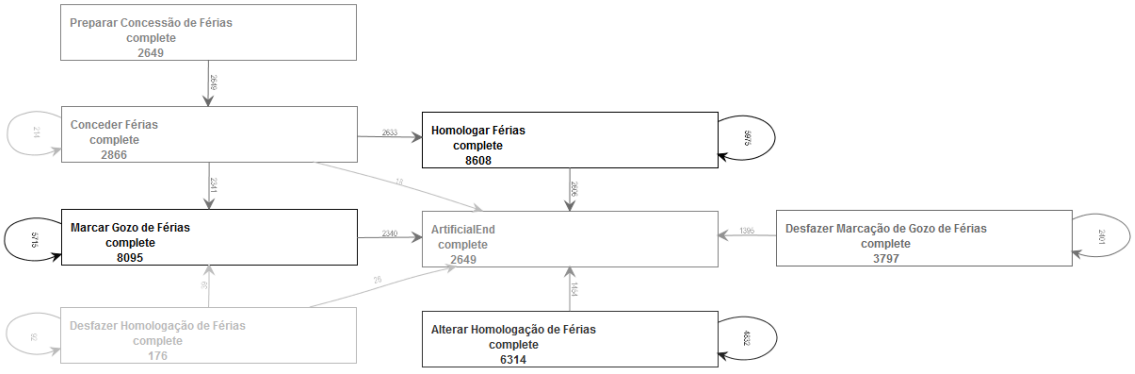

Figura 5.10: Modelo final do processo de férias minerado com o algoritmo *Heuristics Miner*.

Na Figura 5.13 é possível visualizar os fluxos minerados para as quatro variantes do fluxo mais significativas, em termos de quantidade de ocorrências. Do lado esquerdo é possível ver o fluxo das Variantes 1, 2 e 3, que diferem apenas na quantidade de execuções das quatro tarefas envolvidas. Do lado direito está o fluxo da Variante 4, que possui poucas diferenças em comparação com o fluxo das três primeiras.

5.6 Analisar Volume de Retrabalho

Neste momento foi avaliado se uma mesma tarefa está sendo executada diversas vezes em cada *case*, caracterizando retrabalho e, possivelmente, desperdício de tempo dos recursos alocados.

Ao utilizar a rotina Java do Apêndice B, a saída da Listagem 5.2 foi gerada.

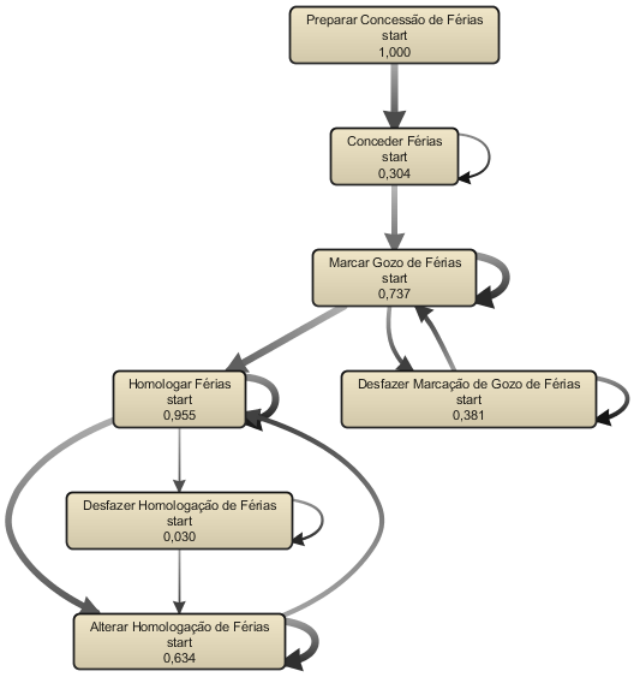

Figura 5.11: Modelo do processo de férias minerado com o algoritmo *Fuzzy*.

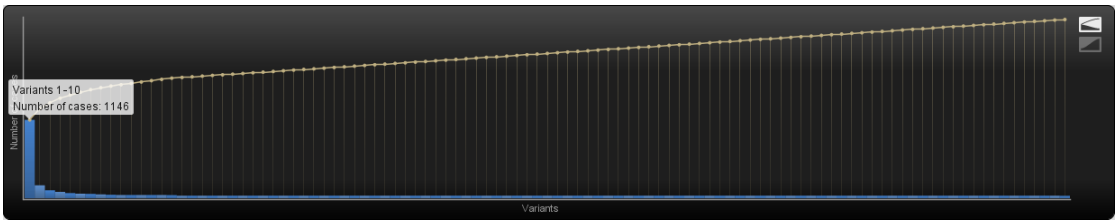

Figura 5.12: Variantes do fluxo.

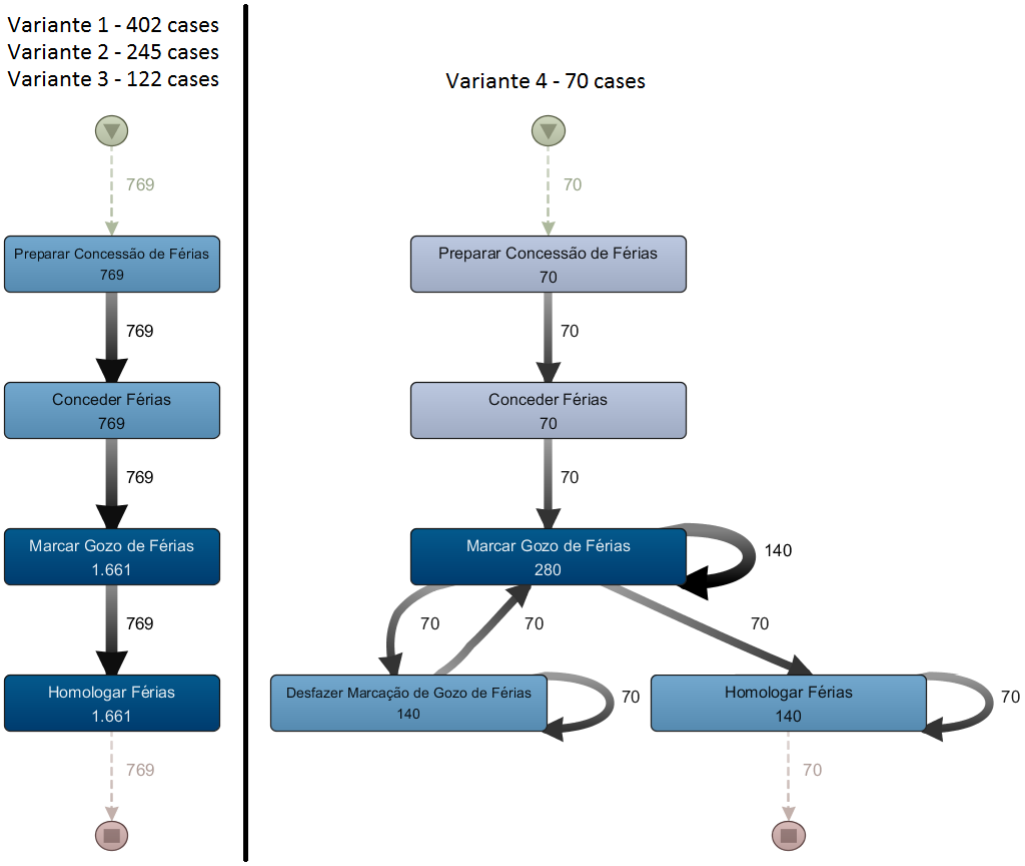

Figura 5.13: Fluxo das quatro variantes mais significativas.

Listagem 5.2: Saída da rotina que contabiliza o fator de recorrência para o *log* do Tribunal Federal

```
Parsing XES file 'ferias.xes' ...

Cases: 2.649

Recurrence Factor
-----------------

Desfazer Homologação de Férias:  2,1 (taskCount=176/casesWithTask=82) [
    ↪ MAX: caseId=2014/0000001837, count=22]
Alterar Homologação de Férias:   4,3 (taskCount=6314/casesWithTask=1482)
    ↪  [MAX: caseId=2015/0000001387, count=51]
Conceder Férias:         1,1 (taskCount=2866/casesWithTask=2649) [MAX:
    ↪ caseId=2015/0000002455, count=2]
Homologar Férias:        3,3 (taskCount=8608/casesWithTask=2632) [MAX:
    ↪ caseId=2014/0000001837, count=28]
Marcar Gozo de Férias:   3,5 (taskCount=8095/casesWithTask=2341) [MAX:
    ↪ caseId=2014/0000003189, count=21]
Desfazer Marcação de Gozo de Férias:     2,7 (taskCount=3797/
    ↪ casesWithTask=1396) [MAX: caseId=2014/0000003189, count=19]
```

A partir das frequências geradas por essa rotina, foi possível notar que a tarefa "Alterar Homologação de Férias" ocorre mais vezes ao longo dos diferentes *cases* do que as outras tarefas, com uma média de 4,3 vezes por *case*. No *case* de maior recorrência dessa tarefa, de ID "2015/0000001387", essa tarefa foi realizada 51 vezes. A segunda tarefa mais executada é a "Marcar Gozo de Férias", ocorrendo, em média, 3,5 vezes por *case*. No *case* de maior recorrência dessa tarefa, de ID "2014/0000003189", essa tarefa foi realizada 21 vezes.

5.7 Analisar Desempenho

A primeira análise de desempenho foi realizada observando o ritmo de início e término dos *cases* ao longo do tempo, representado pelo gráfico *Dotted Chart* gerado pelo ProM. Nele é possível ver de forma superficial e abrangente o ritmo em que as férias foram atendidas ao longo do biênio 2014-2015. A Figura 5.14 retrata o gráfico gerado.

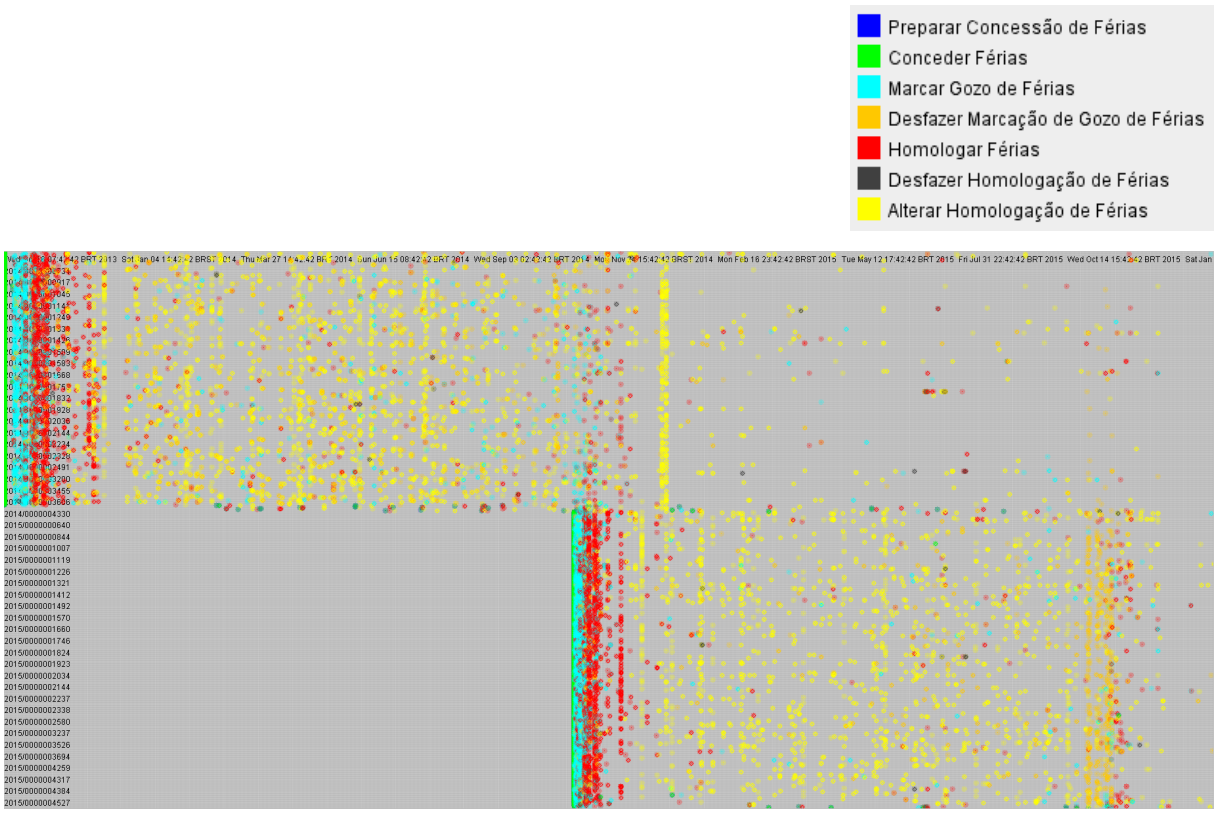

Figura 5.14: *Dotted Chart* gerado pelo ProM, onde os Eixos X e Y representam o mês/ano do evento e o ID do *case*, respectivamente. A cor de cada ponto representa a tarefa realizada, de acordo com a legenda de cores apresentada.

Cada ponto no gráfico representa um evento que ocorreu num determinado momento (Eixo X) e que pertence a um *case* em particular (Eixo Y). A legenda de cores designa a tarefa sendo realizada. É possível notar uma distância de tempo entre os *cases* de cada ano de exercício, sendo que no quadrante superior eles iniciaram em Outubro de 2013, e no quadrante inferior iniciaram em Novembro de 2014.

A tarefa "Preparar Concessão de Férias", de cor azul escuro, ocorre uma só vez, no início de todos os *cases*, sendo que ela tem uma duração muito pequena, ficando quase imperceptível no gráfico. A tarefa "Conceder Férias", de cor verde, ocorre de uma só vez, em lote (cadastro feito através da funcionalidade Gerar Escala de Férias, do SI), logo em seguida à tarefa "Preparar Concessão de Férias", e seguida pelos os pontos azul-claros, da tarefa "Marcar Gozo de Férias". Posteriormente estão os pontos vermelhos, da tarefa "Homologar Férias". Essas três primeiras tarefas ocorrem mais ou menos no período de Outubro a Novembro do ano anterior ao ano de exercício. No restante dos meses existe uma predominância dos pontos amarelos, da tarefa "Alterar Homologação de Férias".

Outra análise de desempenho foi feita através da ferramenta Disco na opção de visualização do processo com informações de tempo total de duração de cada passo do fluxo, conforme visto na Figura 5.15.

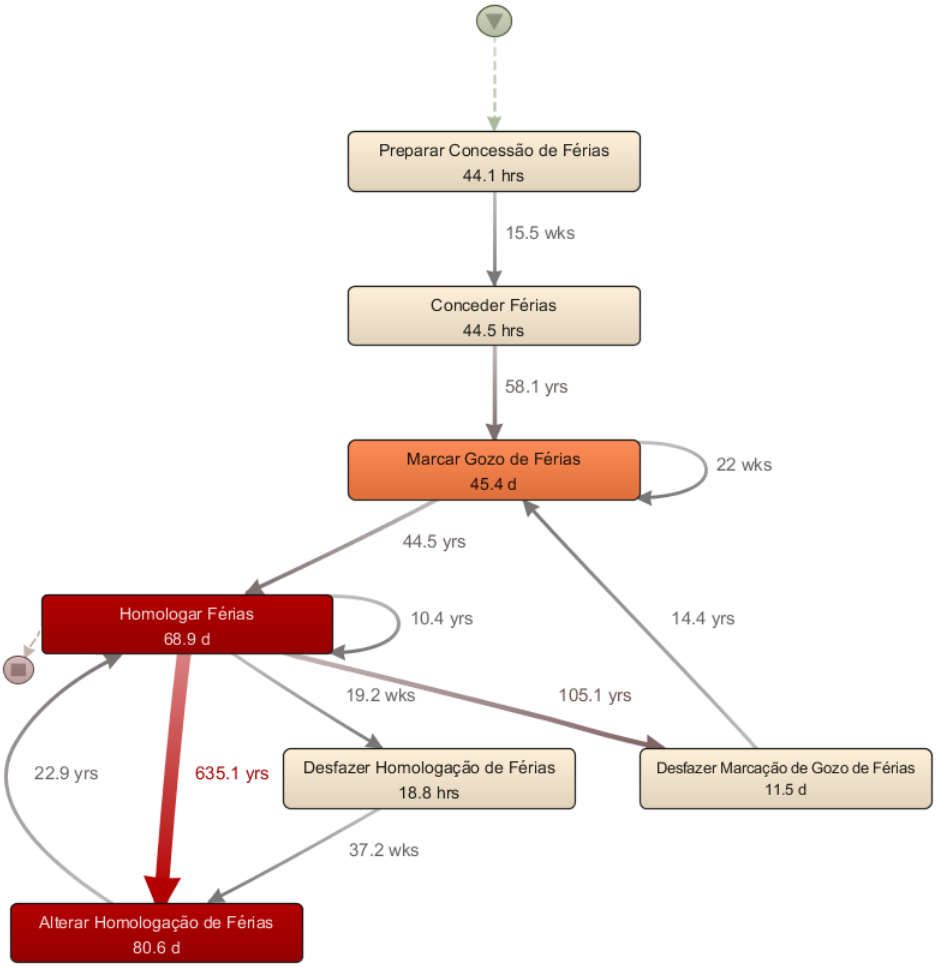

Figura 5.15: Fluxo com o total de duração dos passos.

É possível notar pelo gráfico que algumas tarefas, em particular, se destacaram pelo tempo total de realização em comparação com as demais. Essas tarefas estão marcadas em laranja ou vermelho: "Marcar Gozo de Férias", "Homologar Férias" e "Alterar Homologação de Férias". E a tarefa que consome menos tempo é a "Desfazer Homologação de Férias".

Apesar de algumas transições entre as tarefas apresentarem valores consideravelmente grandes, como a transição da tarefa "Homologar Férias" para a "Alterar Homologação de Férias", esses valores condizem com a realidade. Essa transição, em particular, possui o valor de 635,1 anos, que representa o período total de espera entre todas as homologações e alterações de férias dos servidores no período analisado. Ao analisar essa transição na ferramenta Disco, foi possível descobrir que ela ocorreu 2.452 vezes, com um tempo médio de 3,1 meses $(635,1 * 12/2.452)$ e máximo de 18,6 meses.

5.8 Descobrir Recursos e Papéis Envolvidos

Foi utilizada uma opção na ferramenta Disco para visualizar os diferentes papéis envolvidos e a ocorrência com que eles interagem no processo, conforme ilustrado na Figura 5.16. De acordo com a figura, o papel do analista de RH é o maior envolvido no processo, atuando em 47,05% das atividades realizadas. Em seguida está o Servidor, que realiza 31,24%. O papel do Gestor atua na menor quantidade das atividades, em 21,71%.

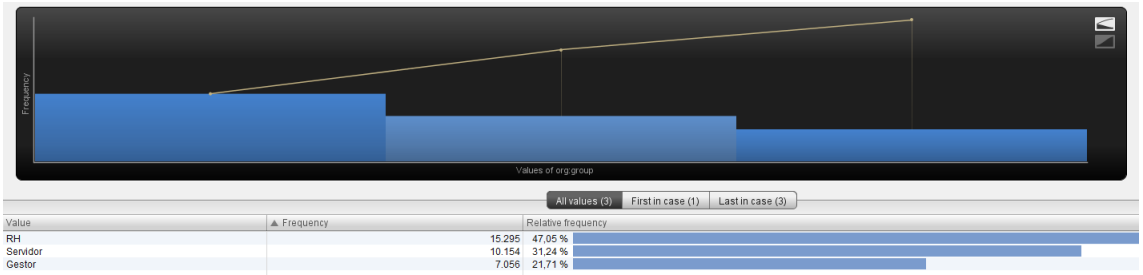

Figura 5.16: Papéis envolvidos e frequências de participação.

5.9 Descobrir a Hierarquia e Caracterizar Papéis Gerenciais e Operacionais

Foi utilizado um gráfico que permite visualizar os recursos de forma hierárquica, gerado pelo ProM utilizando o algoritmo *Role Hierarchy Miner*. Como o gráfico faz uso de uma fonte de texto reduzida, apenas a parte mais relevante do gráfico foi mostrada na Figura 5.17.

Ao acessar esse gráfico pelo ProM, é possível visualizar quais recursos realizam determinada tarefa. A tarefa "Homologar Férias" é uma tarefa gerencial, e está sendo realizada apenas por recursos que possuem o prefixo "Gestor". Já as tarefas "Preparar Concessão de Férias", "Conceder Férias", "Desfazer Homologação de Férias" e "Alterar Homologação de Férias", que também possuem caráter gerencial, são realizadas apenas por recursos que possuem o prefixo "RH". As tarefas "Marcar Gozo de Férias" e "Desfazer Marcação de Gozo de Férias", que são essencialmente atividades operacionais, são realizadas tanto por recursos que possuem o prefixo "Servidor" quanto pelos que possuem prefixo "Gestor".

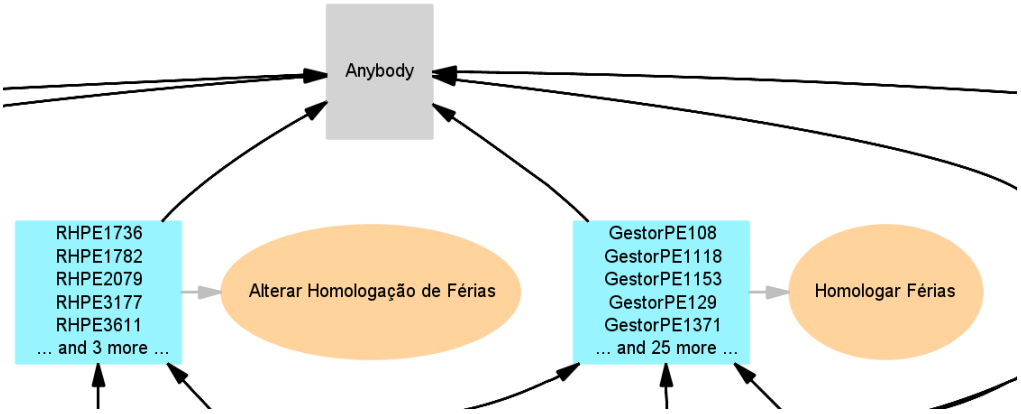

Figura 5.17: Hierarquia de recursos associados com as tarefas.

5.10 Analisar Envolvimento da Gerência nas Atividades Gerenciais e Operacionais

De acordo com o gráfico da Figura 5.16, foi possível levantar as informações descritas na Tabela 5.8. Essa tabela expressa o nível de participação dos papéis gerenciais (Gestor e Analista de RH) nas atividades de caráter gerencial e operacional, conforme classificação definida no Item 5.9.

Tabela 5.8: Papéis e envolvimento nas atividades.

Papel	**Tipo de Atividade**	**Frequência**
Gestor	Gerencial	99%
	Operacional	1%
Analista de RH	Gerencial	100%
	Operacional	0%

Isso significa que, tanto os Gestores quanto os Analistas de RH se dedicam essencialmente a atividades de caráter gerencial (99% e 100%). Apenas os Gestores executam atividades de caráter operacional, mas com um percentual insignificativo (apenas 1%). Portanto, é possível concluir que os papéis gerenciais atuam basicamente nas atividades de caráter gerencial do processo.

5.11 Analisar Sobrecarga de Recursos e Volume de Transferência de Trabalhos

Para efetuar análises de sobrecarga e transferência de trabalho, foi utilizado o algoritmo *Mine for a Working-Together Social Network* na ferramenta ProM. A opção *size by ranking*

desse algoritmo foi marcada, para caracterizar os diferentes papéis numa visão macro do repasse de trabalho dentro do processo de férias. Ao minerar todas as lotações do órgão, o gráfico gerado ficou de difícil entendimento, conforme Figura 5.18.

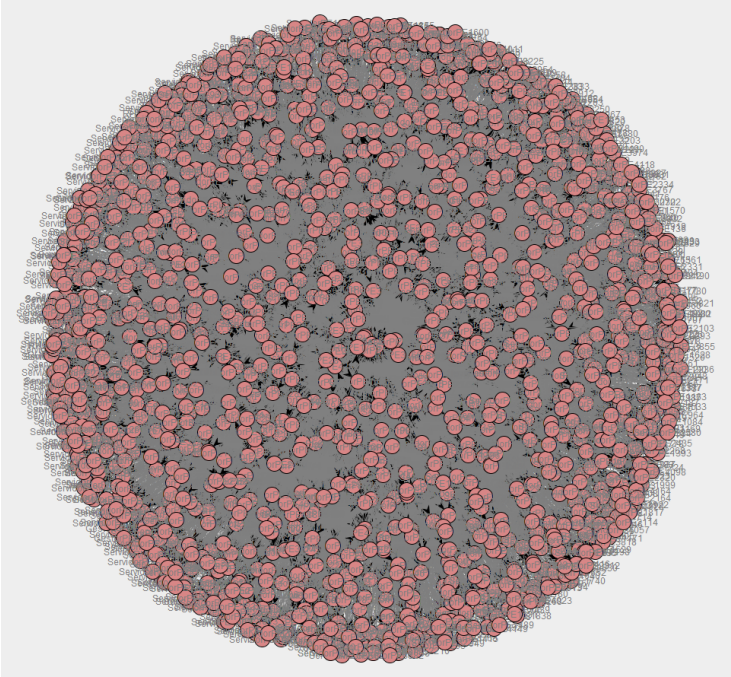

Figura 5.18: Gráfico ilustrativo do repasse de trabalho, gerado pelo *Mine for a Working-Together Social Network*.

Para facilitar o entendimento dessa interação social, os dados minerados foram filtrados para apenas uma lotação, o departamento "Central de Atendimento". O novo gráfico gerado está representado na Figura 5.19. É possível perceber, pelo gráfico, que os recursos de RH são os mais atuantes (como o recurso RH, RHPE3659 e RHPE2105), interagindo com vários servidores e gestores. Além disso, alguns servidores são também significativamente atuantes (como o ServidorPE1359 e o ServidorPE1690), efetuando várias marcações e desmarcações de férias, que são enviadas e retornadas pelos gestores correspondentes ou recursos de RH, apesar de que esse nível de interatividade não era esperado.

5.12 Consolidar e Apresentar Conclusões e Recomendações Finais

No final da análise, as informações foram consolidadas e apresentadas ao cliente. Os problemas inicialmente levantados foram respondidos através dos resultados obtidos. Foi possível entender de uma forma mais detalhada o processo de férias, visualizar o seu fluxo, avaliar aspectos como desempenho do processo e a colaboração dos diferentes recursos envolvidos.

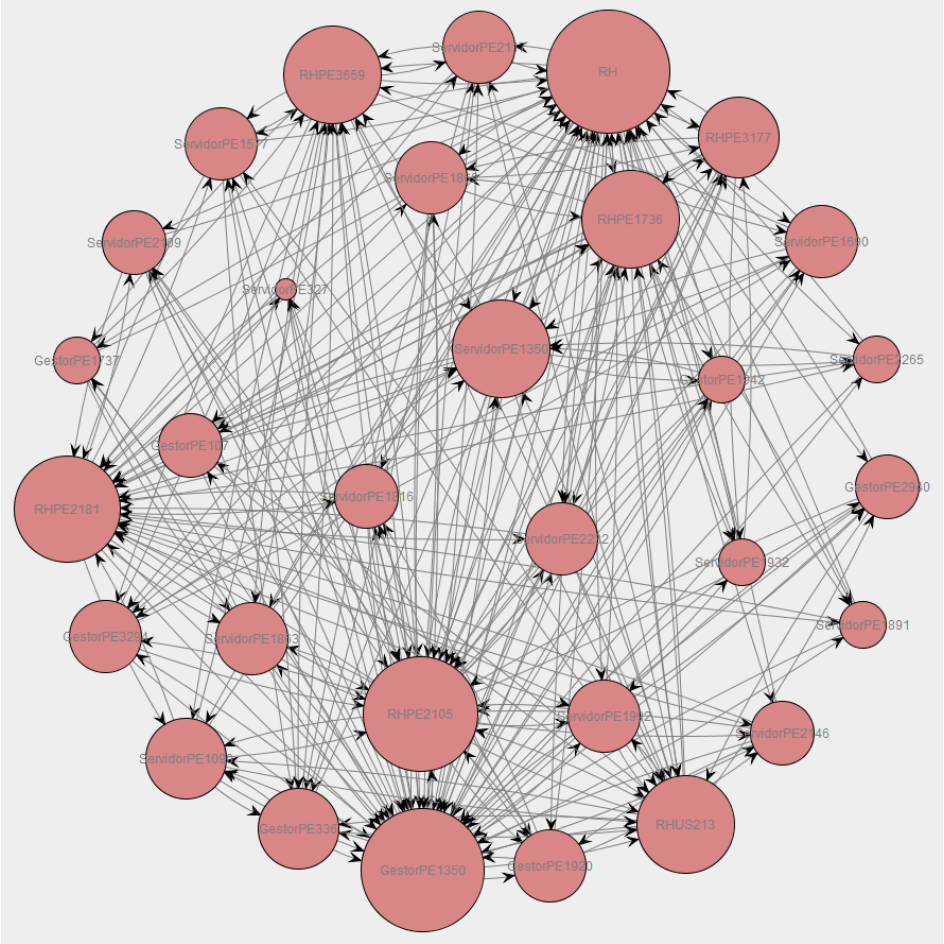

Figura 5.19: Gráfico ilustrativo do repasse de trabalho para apenas um Departamento do órgão.

Resumo das Análises

As perguntas apresentadas na Seção 1.2 foram respondidas conforme segue.

Pergunta 1. É possível reduzir custos na execução dos passos no processo do controle de férias?

A análise realizada no Passo 7 mostrou que um esforço significativo tem sido dedicado à alteração da homologação de férias, além da devida marcação e homologação de férias, conforme Figura 5.20. As três tarefas, juntas, somam 92,4% do tempo total dispendido no processo. Uma forma de reduzir custos seria levantar as causas desse desperdício de recursos e tempo, com o objetivo de incentivar um melhor planejamento na marcação e gozo de férias pelos servidores, para então promover a devida redução de recorrências dessa tarefa a curto e médio prazo.

Por exemplo, pode ser que a causa da constante remarcação das férias seja proveniente da própria coordenação dos trabalhos no órgão, requerendo constantes remarcações de férias para atender a prazos ou metas de alta prioridade. Nesse caso, deve ser averiguada a possibilidade de haver um melhor planejamento das demandas pelos gestores envolvidos.

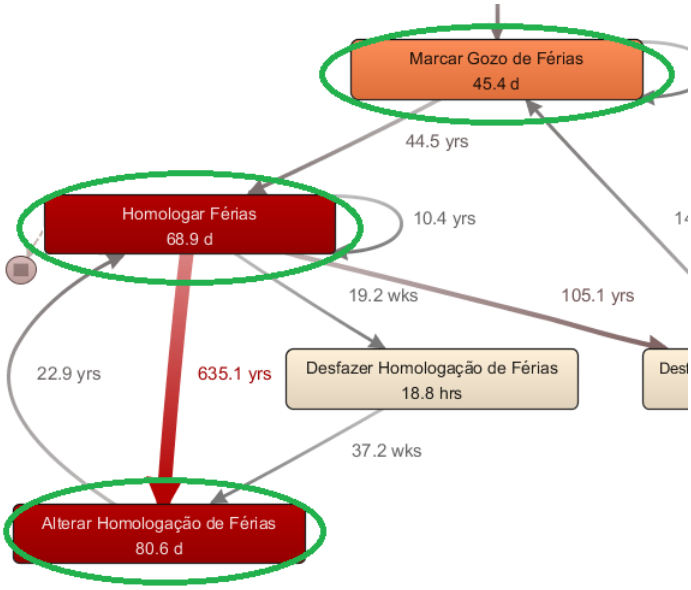

Figura 5.20: Fluxo com os totais mais significativos de duração.

Pode ser que a causa desse desperdício seja uma falta de planejamento pessoal, na qual o servidor muda de idéia com relativa frequência. Uma forma de combater isso é incentivar amplamente no órgão um maior cuidado com essa marcação, e talvez definir regras mais rígidas que limitem o número de remarcações possíveis.

Pode ser também que faltem recursos de RH para atender ao volume de demanda de remarcações de férias. Ou pode ser que o processo de alteração precise ser otimizado, disponibilizando a funcionalidade de remarcação de férias ao servidor também, e não apenas à equipe de RH. Dessa forma, a comunicação e formalização dessas alterações ficará nas mãos de quem está mais informado e interessado nessas mudanças. Essa opção exigiria o desenvolvimento de novas funcionalidades no sistema.

Pergunta 2. Existe volume significativo de retrabalho?

O volume de retrabalho foi observado de forma significativa no Passo 6, sendo que a tarefa "Alteração de Homologação de Férias" se destacou pela grande recorrência nos *cases* (média de 4,3 ocorrências por *case*), conforme observado na Listagem 5.2 (pg. 62). Essa constatação reforça o problema descrito na Pergunta 1.

Pergunta 3. Como o sistema envolvido implementou o fluxo de controle de férias?

O real modelo do fluxo de controle de férias foi minerado no Passo 4, tornando possível entender quais são os passos envolvidos e em que ordem ocorrem. Foi possível perceber pelo fluxo minerado no algoritmo *Fuzzy Miner* que o entendimento inicial do fluxo era diferente, já que existem vários caminhos possíveis de retrocesso no fluxo, e não apenas uma sequência linear, conforme modelado inicialmente. Isso significa que os analistas envolvidos nas reuniões de levantamento do Passo 1 não possuíam uma visão completa e precisa do processo, conforme estava implementado no sistema. A diferença dos dois fluxos está ilustrada na Figura 5.21 (pg. 70).

Pergunta 4. Quais são os reais recursos e papéis envolvidos no controle de férias?

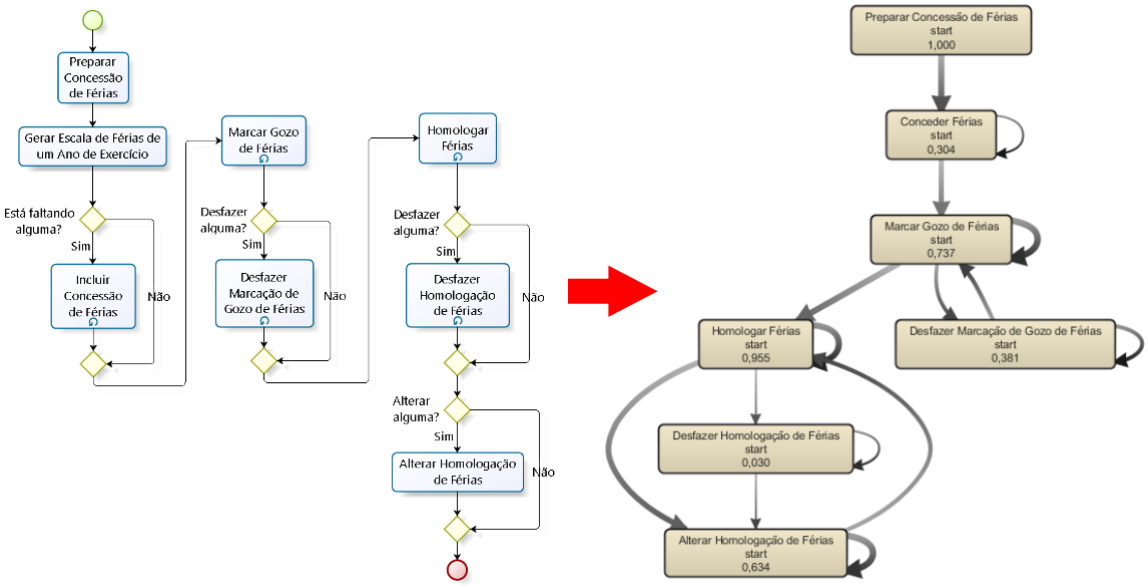

Figura 5.21: Diferenças entre o fluxo esperado e real do processo de controle de férias.

Os recursos e papéis envolvidos no controle de férias foram identificados nos Passos 1, 3 e 9, conforme Tabela 5.5 (pg. 55). O nível de participação de cada um está ilustrado na Figura 5.16 (pg. 65). Aparentemente faltam recursos na equipe de análise de RH, como já sugerido na Pergunta 1, ou o sistema necessita de novas funcionalidades, já que os analistas de RH desempenham um papel central e são os únicos responsáveis por efetuar as alterações de homologação de férias até este momento, apesar da atividade ser recorrente nos *cases*.

Pergunta 5. Qual é o nível de envolvimento dos gerentes no processo?

O papel gerencial do processo de controle de férias está subdividido entre os gestores de área e os analistas de RH, de acordo com o Passo 10. Existe uma participação equilibrada dos dois papéis no processo, sendo que o envolvimento de ambos nas atividades operacionais é praticamente insignificante.

Pergunta 6. Existe sobrecarga de recursos no controle de férias?

De acordo com a análise realizada no Passo 11, os analistas de RH possuem um papel central no processo, e estão mais sobrecarregados que os gestores, indicando um provável gargalo e risco no desempenho do processo.

Neste capítulo foi apresentado um cenário real de aplicação do método proposto. Cada passo do método foi exercitado nesse estudo de caso, sendo descritas as entradas envolvidas, os algoritmos e ferramentas utilizadas, a abordagem de trabalho e o resultado alcançado. Ao final, foram respondidas algumas perguntas que motivaram esse projeto de MP. No próximo capítulo serão feitas algumas conclusões e descritos passos futuros que poderão dar continuidade à esta pesquisa.

Capítulo 6

Conclusões e Trabalhos Futuros

Ao considerar a disciplina de mineração de processos como uma atividade complexa, que envolve um conjunto de atividades, papéis, técnicas e ferramentas, o método utilizado pelo especialista em mineração exerce grande influência no direcionamento dos trabalhos. Este trabalho apresentou um método de MP que consiste de doze passos, sendo uma versão detalhada dos métodos de Bozkaya et al. (2009) e van der Ham (2015). A descrição do método sugere técnicas, algoritmos e ferramentas que podem ser utilizados em casos práticos. A sua aplicação foi ilustrada num caso real de uma empresa pública judicial, com a finalidade dar apoio à tomada de decisões e possibilitar a redução de custos e a otimização do processo analisado.

A análise realizada caracterizou um volume considerável de retrabalho, em especial na tarefa "Alterar Homologação de Férias", executada, em média, 4,3 vezes dentre todas as instâncias do processo de férias. A análise indicou ainda sobrecarga do papel "Analista de RH", atuando em 47,05% das atividades realizadas, apesar de representar apenas 3,45% dos recursos envolvidos. Esses são indicadores de questões a serem analisadas e possivelmente revisadas desse processo para que se conquiste uma execução mais eficiente.

O método proposto possui um nível de generalidade que permite aplicá-lo em outras áreas ou contextos de negócio. No entanto, também possui algumas limitações a serem consideradas. O método não explora a utilização de todas as técnicas, algoritmos e ferramentas disponíveis. Essa restrição faz com que algumas análises mais específicas, como a comparação entre processos similares ou a análise de processos de considerável dinamicidade (que sofrem frequentes alterações no fluxo, ao longo do tempo) não estejam contemplados no escopo de sua utilização.

A comunicação desta pesquisa foi feita através da publicação do artigo Gandulfo et al. (2016) no Simpósio Brasileiro de Sistemas de Informação (SBSI), que é um evento anual promovido pela SBC (Sociedade Brasileira de Computação) em cooperação com a ACM (Association for Computing Machinery), e afiliado à AIS (Association for Information Systems).

A continuidade desta pesquisa poderá ser feita através da exploração e validação de outros métodos existentes de MP. Mesmo que tenham sido propostos para um contexto em particular, podem ser bastante efetivos no apoio à tomada de decisões e otimização dos processos aplicáveis.

Além disso, outras técnicas de análise de processos podem ser acrescentadas ao método proposto, ou explorados diferentes algoritmos de mineração, como os genéticos e

Markovianos, com o objetivo de maximizar os seus ganhos. O método também pode ser complementado através da exploração de outras ferramentas, como a Minit V.2, promovida no desafio BPIC de 2016[1]. Essa ferramenta é capaz de lidar com grandes quantidades de dados de forma eficiente (van der Aalst, 2016), além de permitir a categorização, separação e cálculo de métricas estatísticas para entender e diagnosticar a realidade do processo.

Outra frente de pesquisa que poderá decorrer deste trabalho é a aplicação do método proposto em novos projetos. Por exemplo, o método pode ser aplicado em projetos de MP que não estejam relacionados com o controle de férias, como o atendimento bancário ou gestão de suprimentos. Dessa forma, poderão ser feitas análises e comparações entre os resultados obtidos.

[1] https://www.win.tue.nl/bpi/doku.php?id=2016:challenge

Referências

Melike Bozkaya, Joost Gabriels, and Jan M. van der Werf. Process diagnostics: A method based on process mining. In *International Conference on Information, Process, and Knowledge Management (eKNOW)*, pages 22–27, Eindhoven, The Netherlands, Feb 2009. IEEE. Doi: 10.1109/eKNOW.2009.29. vi, vii, x, 3, 13, 15, 18, 20, 25, 26, 71

Joos C. A. M. Buijs. Mapping data sources to XES in a generic way. diploma thesis, Technische Universiteit Eindhoven, Mar 2010. 2, 5, 13, 28, 30, 33, 34

Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3): 215–249, 1998. 11

Fluxicon. Data requirements for process mining - flux capacitor. https://fluxicon.com/blog/2012/02/data-requirements-for-process-mining/, Fev 2012. Acessado em: 2016-04-11. 6

Rosemary Francisco and Eduardo A. P. Santos. Aplicação da mineração de processos como uma prática para a gestão do conhecimento. *V Workshop on Business Process Management and Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pages 447–484, May 2011. 23, 24

Pablo I. Gandulfo, Celia G. Ralha, Vanessa T. Nunes, and Elaine do N. Coimbra. Redução de custos nas organizações públicas com método de mineração de processos: um estudo de caso no controle de férias. In Claudia Cappelli, Raul Sidnei Wazlawick, Frank Augusto Siqueira, and Patrícia Vilain, editors, *Proceedings of the XII Brazilian Symposium on Information Systems*, pages 589–596, Florianópolis, Brazil, May 2016. SBSI. 71

Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. In *Proceedings of the 5th International Conference on Business Process Management*, BPM'07, pages 328–343, Berlin, Heidelberg, 2007. Springer-Verlag. Doi: 10.1007/978-3-540-75183-0_24. 10, 11

Joachim Herbst and Dimitris Karagiannis. Workflow mining with inwolve. *Computers in Industry*, 53(3):245–264, 2004. 11

Peter T. G. Hornix. Performance analysis of business processes through process mining. diploma thesis, Technische Universiteit Eindhoven, Jan 2007. 5

Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, Jul 2004. Doi: 10.1.1.122.3308. 19

Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems: Managing the Digital Firm Plus MyMISLab with Pearson eText – Access Card Package*. Prentice Hall Press, Upper Saddle River, NJ, USA, 14th edition, 2015. ISBN 013405847X, 9780134058474. 1

Christopher Marshall, Pearl Brereton, and Barbara Kitchenham. Tools to support systematic reviews in software engineering: A feature analysis. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 13:1–13:10, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2476-2. doi: 10.1145/2601248.2601270. URL http://doi.acm.org/10.1145/2601248.2601270. 19

Álvaro J. S. Rebuge. Business process analysis in healthcare environments. Master of science in information systems and computer engineering, Universidade Técnica de Lisboa, Lisboa, May 2012. https://fenix.tecnico.ulisboa.pt/downloadFile/395144221830/dissertacao.pdf. x, 11, 20, 21, 22, 24

Rabia Saylam and Ozgur K. Sahingoz. Process mining in business process management: Concepts and challenges. In *International Conference on Electronics, Computer and Computation (ICECCO)*, pages 131–134, Eindhoven, The Netherlands, Nov 2013. IEEE. Doi: 10.1109/ICECCO.2013.6718246. 2

Igor E. A. Segers. Deloitte enterprise risk services: Investigating the application of process mining for auditing purposes. diploma thesis, Technische Universiteit Eindhoven, Aug 2007. 32

Fernando Szimanski. *Melhoria de Modelos de Processo de Negócio com Mineração de Processos e Simulação Baseada em Agentes*. PhD thesis, Departamento de Ciência da Computação, Universidade de Brasilia, Brasília, DF, Dec 2013. 2, 11

Ashutosh Tiwari, Chris J. Turner, and Basim Majeed. A review of business process mining: State-of-the-art and future trends. *Business Process Management Journal*, 14 (1):5–22, 2008. Doi: 10.1108/14637150810849373. xii, 10, 11

Wil M. P. van der Aalst. Process mining: Discovering and improving spaghetti and lasagna processes. In N. Chawla, I. King, and A. Sperduti, editors, *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 1–7, Paris, France, April 2011a. IEEE. Doi: 10.1007/978-3-319-02922-1_1. 14, 15

Wil M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Process*. Springer-Verlag, Berlin, Heidelberg, first edition, 2011b. x, 9, 14, 15, 26

Wil M. P. van der Aalst. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia Pacific Business Process Management - First Asia Pacific Conference, AP-BPM 2013, Beijing, China, August 29-30, 2013. Selected Papers*, pages 1–22, 2013a. Doi: 10.1007/978-3-319-02922-1_1. 23

Wil M. P. van der Aalst. Process mining: A historical perspective. Process Mining Camp 2013, 2013b. 2

Wil M. P. van der Aalst. *Process Mining - Data Science in Action.* Springer-Verlag, Berlin, Heidelberg, second edition, 2016. x, 2, 6, 12, 38, 40, 41, 72

Wil M. P. van der Aalst and Minseok Song. Mining social networks: Uncovering interaction patterns in business processes. In *International Conference on Business Process Management 2004, Lecture Notes in Computer Science*, volume 3080, pages 244–260, Springer, Berlin, 2004. Springer-Verlag. Doi: 10.1.1.106.2617. 5

Wil M. P. van der Aalst and Kees van Hee. *Workflow Management: Models, Methods and Systems.* The MIT Press, Jan 2012. 6, 9

Wil M. P. van der Aalst, A. J. M. M. (Ton) Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1128–1142, Sep 2004. Doi: 10.1109/TKDE.2004.47. 10, 11

Wil M. P. van der Aalst, Ana Karla A. de Medeiros, and A. J. M. M. (Ton) Weijters. Genetic process mining. In *Proceedings of the 26th International Conference on Applications and Theory of Petri Nets*, ICATPN'05, pages 48–69, Berlin, Heidelberg, 2005. Springer-Verlag. Doi: 10.1007/11494744_5. 11

Wil M. P. van der Aalst, Boudewijn F. van Dongen, Christian W. Günther, R. S. Mans, Ana Karla A. de Medeiros, A. Rozinat, Vladimir Rubin, Minseok Song, H. M. W. (Eric) Verbeek, and A. J. M. M. (Ton) Weijters. Prom 4.0: Comprehensive support for real process analysis. In Jetty Kleijn and Alex Yakovlev, editors, *ICATPN*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, 2007. Doi: 10.1007/978-3-540-73094-1_28. 2

Wil M. P. van der Aalst, Arya Adriansyah, Ana Karla A. de Medeiros, Franco Arcieri, Thomas Baier, and et al. Blickle, Tobias. Process mining manifesto. *International Conference on Business Process Management 2011, Lecture Notes in Business Information Processing*, 99:169–194, 2011. Doi: 10.1016/j.compind.2003.10.001. 5, 8

Ube van der Ham. Benchmarking of five dutch municipalities with process mining techniques reveals opportunities for improvement. Business Process Intelligence Challenge 2015, 2015. vi, vii, x, 21, 23, 24, 25, 26, 71

Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. (Eric) Verbeek, A. J. M. M. (Ton) Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *Proceedings of the 26th International Conference on Applications and Theory of Petri Nets*, ICATPN'05, pages 444–454, Berlin, Heidelberg, 2005. Springer-Verlag. Doi: 10.1007/11494744_25. 12

A. J. M. M. (Ton) Weijters and Ana Karla A. de Medeiros. Process mining with the heuristicsminer algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 2006. Doi: 10.1.1.614.1352. 11

Dianmin Yue, Xiaodan Wu, Haiyan Wang, and Junbo Bai. A review of process mining algorithms. In *International Conference on Business Management and Electronic Information (BMEI)*, pages 181–185, Tianjin, China, May 2011. IEEE. Doi: 10.1109/ICBMEI.2011.5914454. 2

Augusto B. Zamboni, Andre Di Thommazo, Elis C. M. Hernandes, and Sandra C. P. F. Fabbri. Start - uma ferramenta computacional de apoio à revisão sistemática. *Congresso Brasileiro de Software (CBSoft'10)*, Sep 2010. Doi: 10.1007/978-3-540-73094-1_28. 19

Apêndice A

Configurações de Exemplo da Ferramenta XESame para um Banco de Dados MySQL

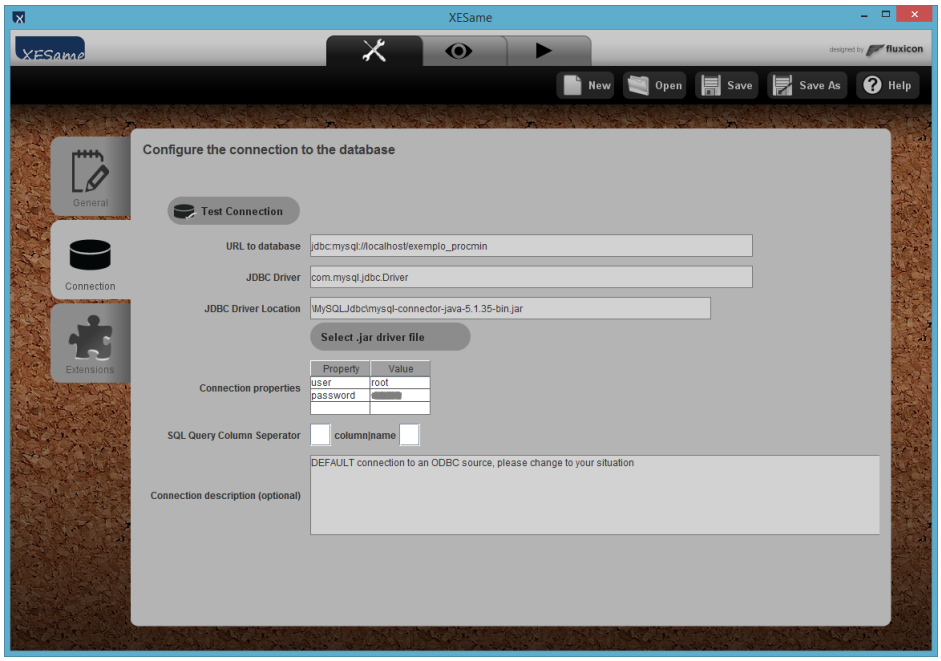

Figura A.1: Configurações de conexão ao MySQL na aba Ferramentas\ *Connection* do XESame.

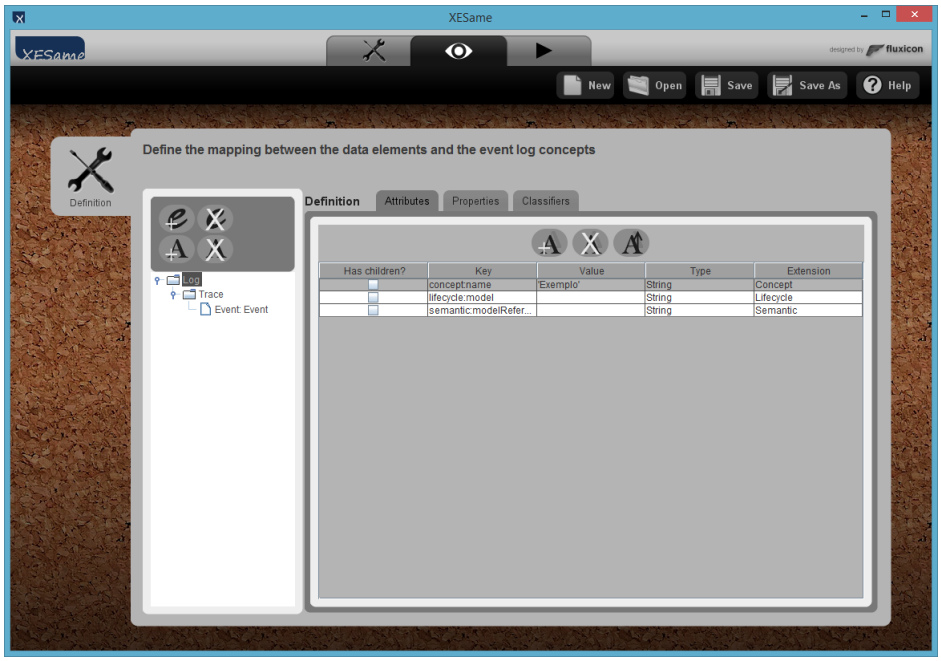

Figura A.2: Configurações de conexão ao MySQL na aba Mapeamento*Definition* do XESame.

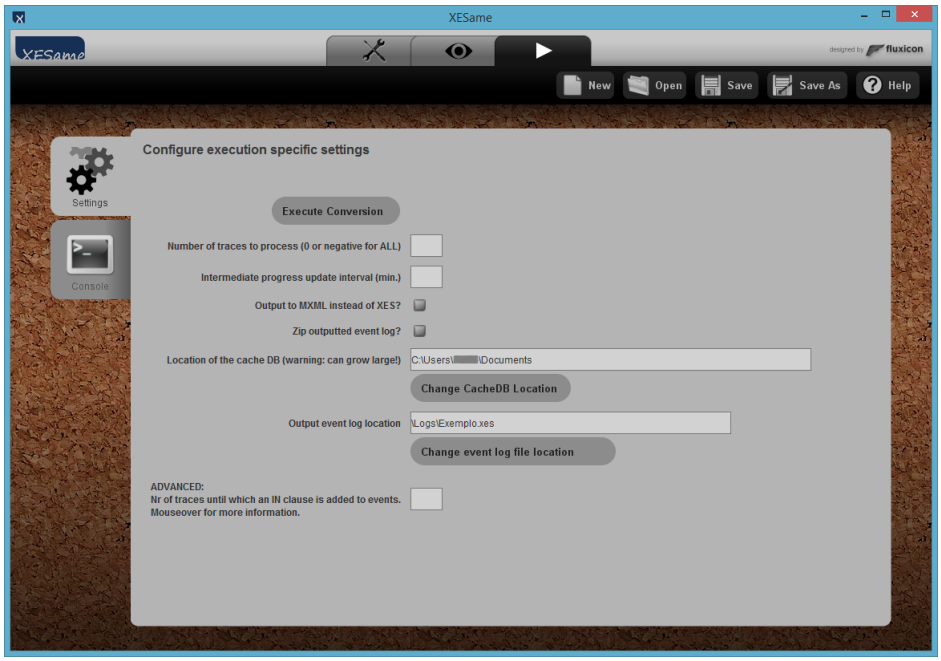

Figura A.3: Configurações de conexão ao MySQL na aba Execução*Settings* do XESame.

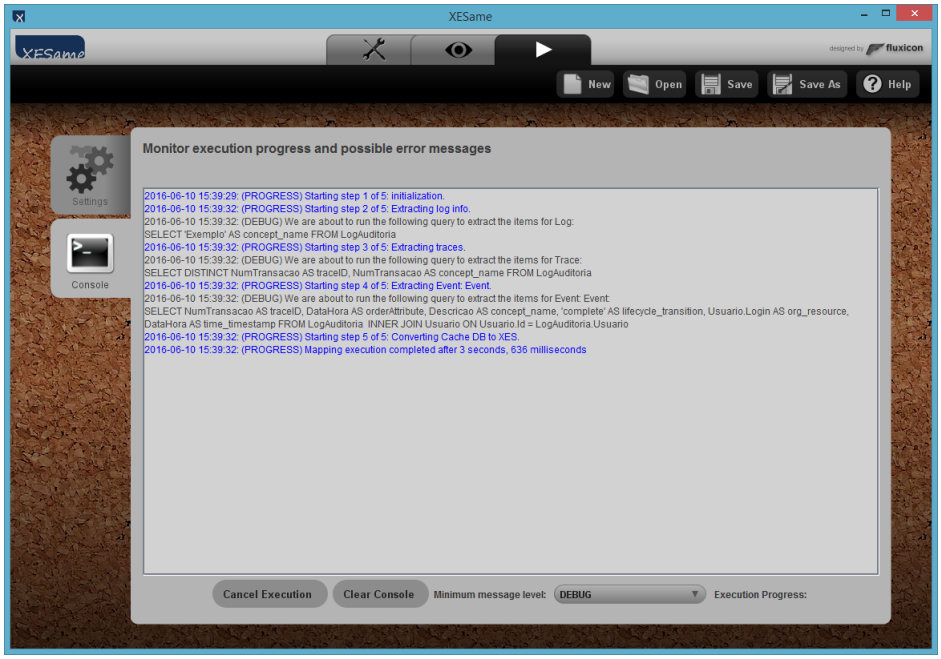

Figura A.4: Configurações de conexão ao MySQL na aba Execução*Console* do XESame.

Apêndice B

Programa 'Rework Counter'

Listagem B.1: Arquivo Main.java

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.math.BigDecimal;
import java.math.MathContext;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import org.deckfour.xes.in.XesXmlParser;
import org.deckfour.xes.model.XEvent;
import org.deckfour.xes.model.XLog;
import org.deckfour.xes.model.XTrace;

/*
 * Main class
 */
public class Main {
  public static String DELIMITER = ";";

  /*
   * Entry point
   */
  public static void main(String[] args) {
    String xesFilePath, caseId, taskName, id, currentTaskName;
    int caseCount, eventCount, casesWithTask, tasks;
    HashMap<String,Integer> tasksPerCaseCounter, casesWithTaskCounter
        ↪ , tasksCounter;
    File xesFile, logFile;
    FileWriter logWriter;
    List<XLog> xesLogs;
    XTrace xesCase;
    XEvent xesEvent;
    Integer counter;
```

```
try {
  // Check program call
  if (args.length != 1) {
    System.out.println("Usage: java Main [xes file]");
    return;
  }

  // Initialize variables
  xesFilePath = args[0];
  caseCount = 0;
  eventCount = 0;
  tasksPerCaseCounter = new HashMap<String,Integer>();
  casesWithTaskCounter = new HashMap<String,Integer>();
  tasksCounter = new HashMap<String,Integer>();

  // Check if XES file exists
  xesFile = new File(xesFilePath);
  if (!xesFile.exists()) {
    System.out.println("XES file '" + xesFilePath + "' not found"
      ↪ );
    return;
  }

  // Program log
  logFile = new File(xesFile.getName().split("\\.")[0] + ".log");
  if (logFile.exists())
    logFile.delete();
  logFile.createNewFile();
  logWriter = new FileWriter(logFile);

  // Parse XES file
  logMessage(logWriter, "Parsing XES file '" + xesFilePath + "'
    ↪ ...\n\n");
  xesLogs = new XesXmlParser().parse(xesFile);

  // XES processing
  for (int i = 0; i < xesLogs.size(); i++) {
    for (int j = 0; j < xesLogs.get(i).size(); j++) {
      xesCase = xesLogs.get(i).get(j);
      caseId = xesCase.getAttributes().get("concept:name").
        ↪ toString().replaceAll(DELIMITER, "");
      caseCount++;

      for (int k = 0; k < xesLogs.get(i).get(j).size(); k++) {
        xesEvent = xesLogs.get(i).get(j).get(k);

        if (TransitionStatus.findByValue(xesEvent.getAttributes()
          ↪ .get("lifecycle:transition").
```

```
            toString().toLowerCase()) == TransitionStatus.
                ↪ complete) {
          taskName = xesEvent.getAttributes().get("concept:name")
              ↪ .toString().replaceAll(DELIMITER, "");

          // Increment case counters
          if (!tasksPerCaseCounter.containsKey(caseId + DELIMITER
              ↪ + taskName)) {
            counter = casesWithTaskCounter.get(taskName);
            if (counter == null)
              counter = 0;
            counter++;
            casesWithTaskCounter.put(taskName, counter);
          }

          // Increment task counter
          counter = tasksCounter.get(taskName);
          if (counter == null)
            counter = 0;
          counter++;
          tasksCounter.put(taskName, counter);

          // Increment task per case counter
          counter = tasksPerCaseCounter.get(caseId + DELIMITER +
              ↪ taskName);
          if (counter == null)
            counter = 0;
          counter++;
          tasksPerCaseCounter.put(caseId + DELIMITER + taskName,
              ↪ counter);
        }

        // Print events processed
        eventCount++;
        if (eventCount % 1000 == 0)
          System.out.print("\r Events processed: " + eventCount)
              ↪ ;
      }
    }
  }
  System.out.print("\n\n");

  // Print case count
  logMessage(logWriter, "Cases: " + new DecimalFormat("#,##0").
      ↪ format(caseCount) + "\n\n");

  // Print statistics
  logMessage(logWriter, "Recurrence Factor\n");
  logMessage(logWriter, "-----------------\n\n");
```

```
      for (Iterator<String> i = casesWithTaskCounter.keySet().
         ↪ iterator(); i.hasNext(); ) {
        taskName = i.next();
        casesWithTask = casesWithTaskCounter.get(taskName);
        tasks = tasksCounter.get(taskName);

        caseId = "";
        counter = 0;
        for (Iterator<String> j = tasksPerCaseCounter.keySet().
           ↪ iterator(); j.hasNext(); ) {
          id = j.next();
          currentTaskName = id.split(DELIMITER)[1];

          if (currentTaskName.equals(taskName) && (
             ↪ tasksPerCaseCounter.get(id) > counter)) {
            caseId = id.split(DELIMITER)[0];
                counter = tasksPerCaseCounter.get(id);
          }
        }

        if (tasks > casesWithTask)
          logMessage(logWriter, taskName + ":\t " +
            new DecimalFormat("#,##0.##").format(new BigDecimal(
              (double) tasks / casesWithTask).round(new MathContext
                 ↪ (2))) + " " +
            "(taskCount=" + tasks + "/casesWithTask=" + casesWithTask
               ↪  + ") " +
            "[MAX: caseId=" + caseId + ", count=" + new DecimalFormat
               ↪ ("#,##0").format(counter) + "]\n");
      }

      // Close log
      logWriter.close();
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  }

  /*
   * Print a message in STDOUT and log it
   */
  public static void logMessage(FileWriter logWriter, String message)
     ↪  throws IOException {
    System.out.print(message);
    logWriter.write(message);
  }
}
```

Apêndice C

Programa 'Conversor CSV To XES'

Listagem C.1: Arquivo Main.java

```
import java.io.IOException;
import java.net.URISyntaxException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Hashtable;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

public class Main {
  private static Hashtable<String,Object> otherAttrs = null;

  public static void main(String[] args) {
    try {
      if (args.length != 7) {
        System.out.println("Usage: java Main [hsqldb name] [import
            ↪ csv files=1/0] [csv directory] [csv files] [csv index
            ↪ fields] [sql extended setup] [sql query file]");
        return;
      }

      HsqlDb hsqlDb = new HsqlDb(args[0]);
      //MySQLDb mySQLDb = new MySQLDb(args[0]);

      if (args[1].equals("1")) {
        String[] csvNames = args[3].split(",");
        String[] csvIndexFieldNums = args[4].split(",");

        CsvQuery csvQuery;
```

```
  for (int i = 0; i < csvNames.length; i++) {
    csvQuery = new CsvQuery(args[2], csvNames[i]);
    try {
      DbUtil.runSqlCommand(DbUtil.getDropTableSqlFromRs(
          ↪ csvQuery.getRs(), hsqlDb.getConn()),
        hsqlDb.getConn());
    }
    catch (SQLException e) {
    }
    DbUtil.runSqlCommand(DbUtil.getCreateTableSqlFromRs(
        ↪ csvQuery.getRs(), hsqlDb.getConn()),
      hsqlDb.getConn());
    try {
      DbUtil.runSqlCommand(DbUtil.getDropIndexSqlFromRs(
          ↪ csvQuery.getRs(), csvIndexFieldNums[i]),
        hsqlDb.getConn());
    }
    catch (SQLException e) {
    }
    DbUtil.runSqlCommand(DbUtil.getCreateIndexSqlFromRs(
        ↪ csvQuery.getRs(),
      csvIndexFieldNums[i], hsqlDb.getConn()), hsqlDb.getConn()
          ↪ );
    DbUtil.insertRecordsFromRs(csvQuery.getRs(), hsqlDb.getConn
        ↪ ());
    csvQuery.close();
  }

  if (DbUtil.isMySQL(hsqlDb.getConn())) {
    String sqlExtSetup = DbUtil.correctDelimitedSql(FileUtil.
        ↪ readFile(args[5]), hsqlDb.getConn());
    String[] sqlsExtSetup = sqlExtSetup.split(";");
    for (int i = 0; i < sqlsExtSetup.length; i++)
      if (sqlsExtSetup[i].trim().length() > 0)
        DbUtil.runSqlCommand(sqlsExtSetup[i], hsqlDb.getConn())
            ↪ ;
  }
}

XesWriter xesWriter;
xesWriter = new XesWriter(args[0]);

PreparedStatement stmt = hsqlDb.getConn().prepareStatement(
  DbUtil.correctDelimitedSql(FileUtil.readFile(args[6]), hsqlDb
      ↪ .getConn()));
ResultSet rs = stmt.executeQuery();

System.out.println("Creating␣XES␣file␣'" + args[0] + "'.xes␣...
    ↪ ");
```

```
int count = 0;
boolean started = false;
String caseId = "";
String activity = "", activityOld = "";
Date timestamp = null, timestampOld = null;
String idResource = "", roleResourceOld = "", idResourceOld = "
    ↪ ";
Hashtable<String,Object> otherAttrsOld = null;
otherAttrs = new Hashtable<String,Object>();
otherAttrsOld = new Hashtable<String,Object>();
while (rs.next()) {
  if (!caseId.equals(rs.getString("CaseId"))) {
    started = true;
    caseId = rs.getString("CaseId");
    activity = rs.getString("Activity");
    timestamp = getParsedDate(rs.getString("Timestamp"), "yyyy/
        ↪ MM/dd␣HH:mm:ss");
  }

  if ((!started) && getParsedDate(rs.getString("Timestamp"), "
      ↪ yyyy/MM/dd␣HH:mm:ss").getTime() - timestamp.getTime()
      ↪ <= 0) {
    timestamp = new Date(timestamp.getTime() + (10 * 1000));
  }
  else {
    timestamp = getParsedDate(rs.getString("Timestamp"), "yyyy/
        ↪ MM/dd␣HH:mm:ss");
  }

  if (timestampOld != null) {
    if ((timestamp.getTime() - timestampOld.getTime()) >
        ↪ getSpentTime(activityOld)) {
      timestampOld = new Date(timestampOld.getTime() +
          ↪ getSpentTime(activityOld));
    }
    else {
      timestampOld = new Date(timestamp.getTime() - 1000);
    }

    xesWriter.addEventToCurrentTrace(activityOld, timestampOld,
        ↪  TransitionStatus.complete,
      roleResourceOld, idResourceOld, otherAttrsOld);
  }

  if (started) {
    xesWriter.startTrace(caseId);
  }
```

```
    idResource = rs.getString("IdResource");
    if (idResource.length() == 0) idResource = "(desconhecido)";

    otherAttrs.clear();
    otherAttrs.put("referenceYear", rs.getString("ReferenceYear")
        ↪ );
    otherAttrs.put("referenceLocation", rs.getString("
        ↪ ReferenceLocation"));
    otherAttrs.put("referenceEmployee", rs.getString("
        ↪ ReferenceEmployee"));
    otherAttrs.put("additionalInfo", rs.getString("AdditionalInfo
        ↪ "));

    xesWriter.addEventToCurrentTrace(rs.getString("Activity"),
        ↪ timestamp, TransitionStatus.start,
      rs.getString("RoleResource"), idResource, otherAttrs);

    activityOld = rs.getString("Activity");
    timestampOld = timestamp;
    roleResourceOld = rs.getString("RoleResource");
    idResourceOld = idResource;
    otherAttrsOld.clear();
    otherAttrsOld.putAll(otherAttrs);

    started = false;
    activity = rs.getString("Activity");

    count++;
    if (count % 1000 == 0)
      System.out.print("\r- Count: " + count);
  }
  System.out.println();

  if (timestampOld != null) {
    if ((timestamp.getTime() - timestampOld.getTime()) >
        ↪ getSpentTime(activityOld)) {
      timestampOld = new Date(timestampOld.getTime() +
          ↪ getSpentTime(activityOld));
    }
    else {
      timestampOld = new Date(timestamp.getTime() - 1000);
    }

    xesWriter.addEventToCurrentTrace(activityOld, timestampOld,
        ↪ TransitionStatus.complete,
      roleResourceOld, idResourceOld, otherAttrsOld);
  }

  xesWriter.writeToFile();
```

```
      hsqlDb.close();
    }
    catch (ClassNotFoundException | IOException | SQLException |
       ↪ ParserConfigurationException |
         SAXException | URISyntaxException | ParseException e) {
      e.printStackTrace();
    }
  }

  public static int getSpentTime(String activity) {
    if (activity.equals("Preparar Concessão de Férias"))
      return 1 * 60 * 1000;
    else if (activity.equals("Conceder Férias"))
      return 1 * 60 * 1000;
    else if (activity.equals("Marcar Gozo de Férias"))
      return 20 * 60 * 1000;
    else if (activity.equals("Desfazer Marcação de Gozo de Férias"))
      return 20 * 60 * 1000;
    else if (activity.equals("Homologar Férias"))
      return 30 * 60 * 1000;
    else if (activity.equals("Alterar Homologação de Férias"))
      return 60 * 60 * 1000;
    else if (activity.equals("Desfazer Homologação de Férias"))
      return 30 * 60 * 1000;

    return 0;
  }

  public static Date getParsedDate(String dateTime, String format)
     ↪ throws ParseException {
    if ((dateTime == null) || (dateTime.length() == 0))
      return null;

    dateTime = dateTime.replaceAll("-", "/");
    if (dateTime.split(":").length == 2)
      dateTime = dateTime + ":00";

    return new SimpleDateFormat(format).parse(dateTime);
  }

  public static Object convertNull(Object obj) {
    if (obj == null)
      return "";

    return obj;
  }
}
```

Listagem C.2: Arquivo HsqlDb.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class HsqlDb {
  Connection conn;

  public HsqlDb(String dbName) throws ClassNotFoundException,
     ↪ SQLException {
    Class.forName("org.hsqldb.jdbc.JDBCDriver");
    conn = DriverManager.getConnection("jdbc:hsqldb:file:" + dbName,
       ↪ "sa", "");
  }

  public void close() throws SQLException {
    conn.close();
  }

  public Connection getConn() {
    return conn;
  }
}
```

Listagem C.3: Arquivo MySQLDb.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySQLDb {
  Connection conn;

  public MySQLDb(String dbName) throws ClassNotFoundException,
     ↪ SQLException {
    Class.forName("com.mysql.jdbc.Driver");
    conn = DriverManager.getConnection("jdbc:mysql://localhost/" +
       ↪ dbName, "root", "admin");
  }

  public void close() throws SQLException {
    conn.close();
  }

  public Connection getConn() {
    return conn;
  }
}
```

Listagem C.4: Arquivo CsvQuery.java

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Properties;

public class CsvQuery {
  String csvName;
  Connection conn;
  PreparedStatement stmt;
  ResultSet rs;

  public CsvQuery(String dir, String csvName) throws
    ↪ ClassNotFoundException, IOException, SQLException {
    this(dir, csvName, "SELECT * FROM \"" + csvName + "\"");
  }

  public CsvQuery(String dir, String csvName, String sql) throws
    ↪ ClassNotFoundException, IOException, SQLException {
    this.csvName = csvName;

    InputStream fis = new FileInputStream(System.getProperty("user.
      ↪ dir") + "/" + csvName + ".csv");
    InputStreamReader isr = new InputStreamReader(fis);
    BufferedReader br = new BufferedReader(isr);
    String headerLine = br.readLine();
    br.close();
    isr.close();
    fis.close();

    Properties props = new Properties();
    props.put("separator", ";");
    props.put("defectiveHeaders", new Boolean(true));
    props.put("suppressHeaders", "true");
    props.put("headerline", headerLine);

    Class.forName("org.relique.jdbc.csv.CsvDriver");
    if (dir.startsWith("./"))
      dir = System.getProperty("user.dir") + "/" + dir.substring(2);
    conn = DriverManager.getConnection("jdbc:relique:csv:" + dir,
      ↪ props);
    stmt = getConn().prepareStatement(sql);
```

```
    rs = getStmt().executeQuery();
    getRs().next();
  }

  public void close() throws SQLException {
    rs.close();
    stmt.close();
    conn.close();
  }

  public String getCsvName() {
    return csvName;
  }

  public Connection getConn() {
    return conn;
  }

  public PreparedStatement getStmt() {
    return stmt;
  }

  public ResultSet getRs() {
    return rs;
  }
}
```

Listagem C.5: Arquivo DbUtil.java

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class DbUtil {
  public static String correctDelimitedSql(String sql, Connection
    ↪ conn) throws SQLException {
    return sql.replaceAll("\\^", getDelimiter(conn));
  }

  public static void runSqlCommand(String sqlCommand, Connection conn
    ↪ ) throws SQLException {
    if (sqlCommand.length() == 0)
      return;

    System.out.println(sqlCommand);

    Statement stmt = conn.createStatement();
```

```
    if (isMySQL(conn)) {
      stmt.execute("SET AUTOCOMMIT = 0");
    }
    else if (isHsql(conn)) {
      stmt.execute("SET AUTOCOMMIT false");
    }
    stmt.execute(sqlCommand);
    stmt.execute("COMMIT");
    stmt.close();
  }

  public static boolean insertRecordsFromRs(ResultSet srcRs,
     ↪ Connection conn) throws SQLException {
    ResultSetMetaData srcRsMetaData = srcRs.getMetaData();
    if (srcRsMetaData.getColumnCount() == 0)
      return false;

    System.out.println("Inserting records on table '" + srcRsMetaData
       ↪ .getTableName(1) + "' ...");

    PreparedStatement destPstmt = null;
    Statement stmt = null;
    ResultSet destRs = null;

    if (isMySQL(conn)) {
      StringBuilder sql = new StringBuilder(1024);
      sql.append("INSERT INTO ").append(getDelimitedName(
         ↪ srcRsMetaData.getTableName(1), conn)).
        append(" (").append(getDelimitedName("_Id", conn));
      for (int i = 1; i <= srcRsMetaData.getColumnCount(); i++) {
        if (srcRsMetaData.getColumnLabel(i).length() == 0)
          break;
        sql.append(", ").append(getDelimitedName(srcRsMetaData.
           ↪ getColumnLabel(i), conn));
      }
      sql.append(") VALUES (?");
      for (int i = 1; i <= srcRsMetaData.getColumnCount(); i++) {
        if (srcRsMetaData.getColumnLabel(i).length() == 0)
          break;
        sql.append(", ?");
      }
      sql.append(")");
      destPstmt = conn.prepareStatement(sql.toString());

      stmt = conn.createStatement();
      stmt.execute("SET AUTOCOMMIT = 0");
    }
    else if (isHsql(conn)) {
```

```
    stmt = conn.createStatement(ResultSet.TYPE_FORWARD_ONLY,
        ↪ ResultSet.CONCUR_UPDATABLE);
    destRs = stmt.executeQuery("SELECT * FROM " + getDelimitedName(
        ↪ srcRsMetaData.getTableName(1), conn));
    if (destRs.getConcurrency() == ResultSet.CONCUR_READ_ONLY) {
      System.out.println("Não foi possível abrir um cursor de
          ↪ atualização!\n");
      destRs.close();
      stmt.close();

      return false;
    }

    stmt.execute("SET AUTOCOMMIT false");
  }

  int count = 0;
  while (srcRs.next()) {
    if (isMySQL(conn)) {
      destPstmt.setString(1, String.valueOf(count + 1));
      for (int i = 1; i <= srcRsMetaData.getColumnCount(); i++) {
        if (srcRsMetaData.getColumnLabel(i).length() == 0)
          break;
        destPstmt.setString(i + 1, srcRs.getString(srcRsMetaData.
            ↪ getColumnLabel(i)));
      }
      destPstmt.executeUpdate();
    }
    else if (isHsql(conn)) {
      destRs.moveToInsertRow();
      destRs.updateString("_Id", String.valueOf(count + 1));
      for (int i = 1; i <= srcRsMetaData.getColumnCount(); i++) {
        if (srcRsMetaData.getColumnLabel(i).length() == 0)
          break;
        destRs.updateString(srcRsMetaData.getColumnLabel(i), srcRs.
            ↪ getString(srcRsMetaData.getColumnLabel(i)));
      }
      destRs.insertRow();
      destRs.moveToCurrentRow();
    }

    count++;
    if (count % 1000 == 0)
      System.out.print("\r- Count: " + count);
  }
  System.out.println();

  if (isMySQL(conn)) {
    destPstmt.close();
```

```
    }
    else {
      destRs.close();
    }
    stmt.execute("COMMIT");
    stmt.close();

    System.out.println("␣done.");

    return true;
  }

  public static String getCreateIndexSqlFromRs(ResultSet rs, String
     ↪ indexFieldNum, Connection conn) throws SQLException {
    StringBuilder sb = new StringBuilder(1024);

    ResultSetMetaData rsMetaData = rs.getMetaData();
    if (rsMetaData.getColumnCount() == 0)
      return sb.toString();

    String indexField = "";
    for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
      if ((indexFieldNum.length() > 0) && (i == Integer.valueOf(
         ↪ indexFieldNum)))
        indexField = rsMetaData.getColumnLabel(i);
    if (indexField.length() > 0)
      sb.append("CREATE␣INDEX␣idx").append(getIdName(rsMetaData.
         ↪ getTableName(1))).
        append(getIdName(indexField)).append("␣ON␣").
        append(getDelimitedName(rsMetaData.getTableName(1), conn)).
           ↪ append("␣(").
        append(getDelimitedName(indexField, conn)).append(")");

    return sb.toString();
  }

  public static String getDropIndexSqlFromRs(ResultSet rs, String
     ↪ indexFieldNum) throws SQLException {
    StringBuilder sb = new StringBuilder(1024);

    ResultSetMetaData rsMetaData = rs.getMetaData();
    if (rsMetaData.getColumnCount() == 0)
      return sb.toString();

    String indexField = "";
    for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
      if ((indexFieldNum.length() > 0) && (i == Integer.valueOf(
         ↪ indexFieldNum)))
        indexField = rsMetaData.getColumnLabel(i);
```

```
    if (indexField.length() > 0)
      sb.append("DROP INDEX idx").append(getIdName(rsMetaData.
          ↪ getTableName(1))).
        append(getIdName(indexField));

    return sb.toString();
  }

  public static String getCreateTableSqlFromRs(ResultSet rs,
      ↪ Connection conn) throws SQLException {
    StringBuilder sb = new StringBuilder(1024);

    ResultSetMetaData rsMetaData = rs.getMetaData();
    if (rsMetaData.getColumnCount() == 0)
      return sb.toString();

    sb.append("CREATE TABLE ").append(getDelimitedName(rsMetaData.
        ↪ getTableName(1), conn)).
      append(" (" + getDelimitedName("_Id", conn) + " VARCHAR(20), ")
          ↪ ;
    for (int i = 1; i <= rsMetaData.getColumnCount(); i++) {
      if (rsMetaData.getColumnLabel(i).length() == 0)
        break;
      if (i > 1)
        sb.append(", ");
      sb.append(getDelimitedName(rsMetaData.getColumnLabel(i), conn))
          ↪ .append(" ").append(getSqlColumnType(rsMetaData.
          ↪ getColumnTypeName(i)));
      if (getSqlColumnPrecision(rsMetaData.getColumnTypeName(i),
          ↪ rsMetaData.getPrecision(i)) != 0)
        sb.append("(").append(getSqlColumnPrecision(rsMetaData.
            ↪ getColumnTypeName(i), rsMetaData.getPrecision(i))).
            ↪ append(")");
    }
    sb.append(")");

    return sb.toString();
  }

  public static String getDropTableSqlFromRs(ResultSet rs, Connection
      ↪  conn) throws SQLException {
    StringBuilder sb = new StringBuilder(1024);

    ResultSetMetaData rsMetaData = rs.getMetaData();
    if (rsMetaData.getColumnCount() == 0)
      return sb.toString();

    sb.append("DROP TABLE ").append(getDelimitedName(rsMetaData.
        ↪ getTableName(1), conn));
```

```
    return sb.toString();
  }

  public static String getSqlColumnType(String columnType) {
    if (columnType.equals("String"))
      return "VARCHAR";

    return columnType.toUpperCase();
  }

  public static int getSqlColumnPrecision(String columnType, int
     ↪ precision) {
    if (columnType.equals("String") && (precision == 0))
      return 255;

    return precision;
  }

  public static String getDelimitedName(String name, Connection conn)
     ↪  throws SQLException {
    return getDelimiter(conn) + name + getDelimiter(conn);
  }

  public static String getDelimiter(Connection conn) throws
     ↪ SQLException {
    if (isMySQL(conn)) {
      return "`";
    }
    else if (isHsql(conn)) {
      return "\"";
    }

    return null;
  }

  public static String getIdName(String fieldName) {
    return fieldName.replaceAll("[ \\(\\)]", "");
  }

  public static boolean isMySQL(Connection conn) throws SQLException
     ↪ {
    return conn.getMetaData().getDriverName().startsWith("MySQL");
  }

  public static boolean isHsql(Connection conn) throws SQLException {
    return conn.getMetaData().getDriverName().startsWith("HSQL");
  }
}
```

Listagem C.6: Arquivo FileUtil.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class FileUtil {
  public static String readFile(String file) throws IOException {
    StringBuilder stringBuilder = new StringBuilder();
    String ls = System.getProperty("line.separator");

    BufferedReader reader = new BufferedReader(new FileReader(file));
    String line;
    while ((line = reader.readLine()) != null)
      stringBuilder.append(line).append(ls);
    reader.close();

    return stringBuilder.toString();
  }
}
```

Listagem C.7: Arquivo XesWriter.java

```
import org.deckfour.xes.classification.XEventAttributeClassifier;
import org.deckfour.xes.extension.XExtension;
import org.deckfour.xes.extension.XExtensionParser;
import org.deckfour.xes.factory.XFactory;
import org.deckfour.xes.factory.XFactoryRegistry;
import org.deckfour.xes.model.XAttribute;
import org.deckfour.xes.model.XEvent;
import org.deckfour.xes.model.XLog;
import org.deckfour.xes.model.XTrace;
import org.deckfour.xes.out.XesXmlSerializer;
import org.xml.sax.SAXException;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Date;
import java.util.Hashtable;

import javax.xml.parsers.ParserConfigurationException;

public class XesWriter {
  String filename;
  XFactory factory;
```

```
XLog log;
XTrace trace;

public XesWriter(String filename) throws IOException,
    ↪ ParserConfigurationException, SAXException,
    ↪ URISyntaxException {
  XExtensionParser extParser;
  XExtension ext;
  XEventAttributeClassifier attributeClassifier;
  XAttribute attr, childAttr;

  this.filename = filename;
  factory = XFactoryRegistry.instance().currentDefault();
  log = factory.createLog();

  extParser = new XExtensionParser();
  ext = extParser.parse(new URI("http://www.xes-standard.org/
     ↪ concept.xesext"));
  log.getExtensions().add(ext);

  ext = extParser.parse(new URI("http://www.xes-standard.org/
     ↪ lifecycle.xesext"));
  log.getExtensions().add(ext);

  ext = extParser.parse(new URI("http://www.xes-standard.org/org.
     ↪ xesext"));
  log.getExtensions().add(ext);

  ext = extParser.parse(new URI("http://www.xes-standard.org/time.
     ↪ xesext"));
  log.getExtensions().add(ext);

  ext = extParser.parse(new URI("http://www.xes-standard.org/
     ↪ semantic.xesext"));
  log.getExtensions().add(ext);

  attr = factory.createAttributeLiteral("concept:name", "
     ↪ __INVALID__", null);
  log.getGlobalTraceAttributes().add(attr);
  attr = factory.createAttributeLiteral("concept:name", "
     ↪ __INVALID__", null);
  log.getGlobalEventAttributes().add(attr);
  attr = factory.createAttributeLiteral("lifecycle:transition", "
     ↪ complete", null);
  log.getGlobalEventAttributes().add(attr);

  attributeClassifier = new XEventAttributeClassifier("MXML Legacy
     ↪ Classifier", new String[] { "concept:name" });
  log.getClassifiers().add(attributeClassifier);
```

```
    attributeClassifier = new XEventAttributeClassifier("Activity_
        ↪ classifier", new String[] { "concept:name" });
    log.getClassifiers().add(attributeClassifier);
    attributeClassifier = new XEventAttributeClassifier("Event_Name",
        ↪  new String[] { "concept:name" });
    log.getClassifiers().add(attributeClassifier);
    attributeClassifier = new XEventAttributeClassifier("Resource",
        ↪ new String[] { "org:resource" });
    log.getClassifiers().add(attributeClassifier);

    attr = factory.createAttributeLiteral("source", "ConversorCsv2Xes
        ↪ ", null);
    childAttr = factory.createAttributeLiteral("program", "Conversor_
        ↪ Csv_To_Xes", null);
    attr.getAttributes().put(childAttr.getKey(), childAttr);
    log.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeLiteral("concept:name", filename +
        ↪ ".xes", null);
    log.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeLiteral("lifecycle:model", "
        ↪ standard", null);
    log.getAttributes().put(attr.getKey(), attr);
  }

  public void startTrace() {
    startTrace(null);
  }

  public void startTrace(String id) {
    XAttribute attr;

    trace = factory.createTrace();
    log.add(trace);
    if (id != null) {
      attr = factory.createAttributeLiteral("concept:name", id, null)
          ↪ ;
      trace.getAttributes().put(attr.getKey(), attr);
    }
  }

  public void addEventToCurrentTrace(String name, Date timestamp,
      ↪ TransitionStatus transitionStatus, String group, String
      ↪ resource, Hashtable<String, Object> otherAttrs) {
    XEvent event;
    XAttribute attr;

    event = factory.createEvent();
    trace.add(event);
```

```
    attr = factory.createAttributeLiteral("concept:name", name, null)
        ↪ ;
    event.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeTimestamp("time:timestamp",
        ↪ timestamp.getTime(), null);
    event.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeLiteral("lifecycle:transition",
        ↪ transitionStatus.getValue(), null);
    event.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeLiteral("org:group", group, null);
    event.getAttributes().put(attr.getKey(), attr);
    attr = factory.createAttributeLiteral("org:resource", resource,
        ↪ null);
    event.getAttributes().put(attr.getKey(), attr);

    if ((otherAttrs != null) && (otherAttrs.size() > 0))
      for (String key : otherAttrs.keySet())
        if (otherAttrs.get(key) instanceof Boolean) {
          attr = factory.createAttributeBoolean(key, (Boolean)
              ↪ otherAttrs.get(key), null);
          event.getAttributes().put(attr.getKey(), attr);
        }
        else if (otherAttrs.get(key) instanceof Long) {
          attr = factory.createAttributeDiscrete(key, (Long)
              ↪ otherAttrs.get(key), null);
          event.getAttributes().put(attr.getKey(), attr);
        }
        else if (otherAttrs.get(key) instanceof Double) {
          attr = factory.createAttributeContinuous(key, (Double)
              ↪ otherAttrs.get(key), null);
          event.getAttributes().put(attr.getKey(), attr);
        }
        else if (otherAttrs.get(key) instanceof String) {
          attr = factory.createAttributeLiteral(key, (String)
              ↪ otherAttrs.get(key), null);
          event.getAttributes().put(attr.getKey(), attr);
        }
        else if (otherAttrs.get(key) instanceof Date) {
          attr = factory.createAttributeTimestamp(key, (Date)
              ↪ otherAttrs.get(key), null);
          event.getAttributes().put(attr.getKey(), attr);
        }
}

public void writeToFile() throws IOException {
  File file;
  OutputStream os;
  XesXmlSerializer serializer;
```

```
    file = new File(filename + ".xes");
    if (file.exists())
      file.delete();
    file.createNewFile();

    os = new BufferedOutputStream(new FileOutputStream(file));

    serializer = new XesXmlSerializer();
    serializer.serialize(log, os);

    os.close();
  }
}
```

Listagem C.8: Arquivo TransitionStatus.java

```
public enum TransitionStatus {
  schedule, assign, withdraw, reassign, start, suspend, resume,
  pi_abort, ate_abort, complete, autoskip, manualskip, unknown;

  public static TransitionStatus findByValue(String value) {
    return TransitionStatus.valueOf(value);
  }

  TransitionStatus() {
  }

  public String getValue() {
    return toString();
  }
}
```