PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# CONTRIBUIÇÕES NA AVALIAÇÃO DE CONFORMIDADE DE PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE POR MINERAÇÃO DE PROCESSOS

JOHN IGOR BANDEIRA DA CRUZ

Dissertação de Mestrado apresentada como requisito para obtenção do título de Mestre em Ciência da Computação pelo Programa de Pós-graduação da Faculdade de Informática. Área de concentração: Ciência da Computação.

Orientador: Prof. Duncan D. Alcoba Ruiz

Porto Alegre, Brasil
2010

**Dados Internacionais de Catalogação na Publicação (CIP)**

C957c Cruz, John Igor Bandeira da
Contribuições na avaliação de conformidade de processos de desenvolvimento de software por mineração de processos / John Igor Bandeira da Cruz. – Porto Alegre, 2010.
86 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Duncan D. Alcoba Ruiz.

1. Informática. 2. Software – Desenvolvimento. 3. Mineração de Processos. I. Ruiz, Duncan D. Alcoba Ruiz. II. Título.

CDD 005.1

**Ficha Catalográfica elaborada pelo**
**Setor de Tratamento da Informação da BC-PUCRS**

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Contribuições na Avaliação de Conformidade de Processos de Desenvolvimento de Software por Mineração de Processos", apresentada por John Igor Bandeira da Cruz, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas de Informação, aprovada em 16/03/10 pela Comissão Examinadora:

Prof. Dr. Duncan Dubugras Alcoba Ruiz – Orientador — PPGCC/PUCRS

Prof. Dr. Ricardo Melo Bastos – PPGCC/PUCRS

Profa. Dra. Lucinéia Heloisa Thom – UFRGS

Homologada em 25/11/10, conforme Ata No. 23/10. pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
**Campus Central**
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

# Agradecimentos

Agradeço acima de tudo aos meus pais pelo incentivo, sempre procurando mostrar o melhor caminho e dando suporte a todos os desafios que enfrentei até hoje. Sempre estiveram e sei que sempre estarão ao meu lado.

Grande agradecimento à minha namorada por tanto amor, companheirismo e dedicação.

Ao professor Duncan, pela orientação, incentivo, confiança incondicional e amizade.

Ao Convênio PUCRS/HP Brasil por viabilizarem meus estudos e meu crescimento profissional.

A todos que, de uma forma ou de outra, contribuiram para minha formação pessoal e profissional. A todos o que acreditaram e desejaram o melhor pra mim. Aos amigos e parentes, meu sincero agradecimento.

# CONTRIBUIÇÕES NA AVALIAÇÃO DE CONFORMIDADE DE PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE POR MINERAÇÃO DE PROCESSOS

## RESUMO

As empresas de software investem cada vez mais tempo e recursos na melhoria de seus processos. Neste contexto, a mineração de processos pode ser de grande valor servindo como ferramenta para a análise dos processos executados a partir de dados extraídos do próprio ambiente de execução e sistemas de gestão utilizados. Neste trabalho, são discutidos aspectos envolvendo a análise de processos correspondentes ao desenvolvimento de software, utilizando mineração. Foi realizado um estudo de caso exploratório utilizando dados de métricas provenientes de um projeto de manutenção de uma grande operação de software. Os resultados serviram como entrada para uma análise exploratória assim como para uma reflexão acerca das questões que envolvem o processo de descoberta de conhecimento nesse tipo de ambiente. Além disso, o presente trabalho demonstra um cenário prático de aplicação para análise de conformidade aplicando ferramentas de mineração de processos em um processo de software.

**Palavras-chave**: Análise de Conformidade, Processos de Desenvolvimento de Software, Mineração de Processos.

# CONTRIBUITIONS ON CONFORMANCE ANALYSIS ON SOFTWARE DEVELOPMENT PROCESSES USING PROCESS MINING

## ABSTRACT

Software organizations are spending more and more time and resources in their development processes. Conformance analysis is considered an important issue concerning process improvement initiatives. Process mining can be a valuable tool to gather data from event logs produced by management tools for software development environments. Process analysts can use the information obtained to quantify the correspondence between the process models and its real execution. The present work discusses the issues concerning conformance checking and process mining in the context of software development processes. We report an exploratory case study that takes activity records from a maintenance project in a large software operation. The results were useful to examine the issues involved in the knowledge discovery process in this type of environment. In addition, this work presents an application scenario for conformance checking applying process mining tools in a software process.

**Keywords**: Conformance Checking, Process Mining, Software Development.

# Lista de Figuras

# Sumário

# 1. Introdução

Produtos de software e seus respectivos projetos são tarefas complexas atualmente (9). Processos de desenvolvimento de software (PDS) realizados por grandes corporações podem envolver dezenas de colaboradores, executando um grande número de tarefas em uma complexa cadeia de interações. Sendo assim, a definição de modelos de desenvolvimento padronizados e devidamente controlados torna-se fundamental para o sucesso do negócio. Prova da importância da formalização, controle de desempenho e garantia de qualidade de processos é o fato de tais requisitos serem áreas chave em modelos de qualidade como CMMI e ISO (37) (40). Esses modelos de maturidade reúnem requisitos e práticas consideradas essenciais no desenvolvimento e manutenção de produtos de alto nível de qualidade e valor agregado.

O ambiente de processos na engenharia de software costuma ser bem mais dinâmico do que em outras áreas da indústria. Um aspecto importante está relacionado à forma de interação entre os participantes. Projetos de software são altamente cooperativos (52). São geralmente desenvolvidos por equipes multidisciplinares, interagindo constantemente durante todo o processo. Seus colaboradores costumam lidar com múltiplos documentos, onde o compartilhamento de informações e revisões são a tônica. Além disso, PDS têm uma característica intrinsecamente estocástica (43) (14). É difícil fazer generalizações acerca do perfil de execução de projetos possuindo, cada um, metodologias, formas de interação e prazos bem específicos. O projeto de um sistema bancário, por exemplo, exige uma abordagem de desenvolvimento e testes bastante distinta de um portal de informações web onde não se trabalha com informações críticas. Fatores como tipo de projeto, tamanho da equipe, prazo e orçamento acabam afetando consideravelmente o processo adotado pelo projeto.

## 1.1 Motivação

Diante do contexto descrito acima, empresas de software atualmente têm empregado esforço e reconhecem os benefícios da formalização, maior controle e melhoria de seus processos como diferencial competitivo (29). A definição de um conjunto de processos padronizado é um aspecto fundamental nesse contexto. Por outro lado, há a necessidade de garantir a aderência das pessoas aos processos definidos. Normalmente há alguma distância entre esses modelos formais e a realidade de execução. Em médias e grandes empresas, a tarefa de analisar a conformidade entre planejamento e execução dos processos é realizada normalmente por uma equipe independente. Esta busca identificar e comunicar o desempenho dos projetos e desvios de execução do processo. Os resultados

são obtidos geralmente pela realização de auditorias, entrevistas, rastreamento de documentos e análise de métricas.

No entanto, manter a visibilidade e controle adequados em processos de TI não é uma tarefa simples. Diversos autores expressam a necessidade de se obter indicadores apropriados para governança em processos de TI (17) (29) (14). Além disso, o custo para manter uma equipe de qualidade é muitas vezes um fator discutível em projetos de software. Tipicamente, empresas de TI visam máxima produtividade com menor número de pessoas possível, de preferência focadas no projeto e desenvolvimento. Atividades de gestão de qualidade são vistas em certos casos como ”improdutivas”, principalmente por ser mais difícil mensurar suas contribuições sobre o sucesso dos projetos.

Atualmente, técnicas automáticas de extração de conhecimento têm sido aplicadas sobre registros de execução de processos de negócio, buscando obter conhecimento mais preciso e detalhado sobre a sua execução. Tal conjunto de técnicas é conhecido como mineração de processos (4). O objetivo principal é a extração de conhecimento - voltado a processos - sobre conjuntos de eventos. A mineração de processos tem sido explorada amplamente nos últimos anos (6) (15) (20). Na linha de descoberta de processos, conjuntos de eventos são submetidos a algoritmos de mineração com o objetivo de induzir padrões de execução recorrentes. Além disso, a metodologia pode ser aplicada no intuito de verificar se os processos definidos são realmente seguidos por seus participantes. Esse é objetivo da análise de conformidade (*compliance/conformance analysis*).

## 1.2 Contribuições

O presente trabalho apresenta um estudo exploratório envolvendo a aplicação de mineração de processos em dados de execução de PDS. O trabalho explora e discute formas, dificuldades e resultados potenciais relacionados à descoberta de conhecimento sobre processos em um ambiente típico de desenvolvimento de sofware, quando utilizada para verificar a distância entre o que é documentado por uma empresa de TI de grande porte e a realidade. Todo o estudo é baseado em um ambiente real de produção de software, motivado e viabilizado pela parceria de pesquisa PUCRS/HP Brasil. A experiência dentro do grupo de garantia de qualidade da empresa foi um fator fundamental para o desenvolvimento do trabalho. Essa interação permitiu compreender todo o trabalho, benefícios e desafios envolvidos na realização de auditorias de projetos de software. Também foi possível explorar os processos organizacionais e registros de atividades de um dos projetos da empresa parceira, resultando no estudo de caso descrito neste trabalho.

## 1.3 Organização do Texto

O trabalho possui a seguinte estrutura: O capítulo 2 apresenta a fundamentação teórica abordando os temas, qualidade de software, conformidade de processos e mineração de processos. O terceiro capítulo descreve o cenário no qual a pesquisa está inserida, discutindo também o problema abordado no trabalho. O capítulo 4 apresenta o estudo realizado sobre a aplicação de mineração de processos e análise de conformidade em PDS. Primeiramente foram realizados experimentos envolvendo descoberta de modelos de processos sobre dados de execução de um dos projetos da empresa parceira. Além disso, uma solução de verificação de conformidade é explorada em um cenário prático, também utilizando dados reais da empresa. Ao final deste capítulo, foi compilada uma série de sugestões relativas à aplicação de mineração de processos visando comparar formalmente, utilizando algoritmos apropriados, modelo e execução de processos em um ambiente de desenvolvimento de software. O capítulo 5 discute alguns trabalhos relacionados à esta pesquisa. Conclusões e trabalhos futuros são apresentados no capítulo 6.

# 2. Fundamentação teórica

Este capítulo apresenta os principais conceitos relacionados ao tema de pesquisa. Os seguintes assuntos são abordados: (i) qualidade de software, (ii) conformidade de processos, (iii) mineração de processos (iv) redes de Petri e (vi) análise de conformidade utilizando mineração de processos.

## 2.1 Qualidade de software

O trabalho na área de qualidade de software se concentra em desenvolver e aplicar métodos que permitam quantificar o estado atual de um produto ou projeto, com relação ao cumprimento de seus objetivos de negócio. Em projetos de engenharia complexos, como grande parte do desenvolvimento de software atual, é fundamental o estabelecimento de políticas e práticas de garantia de qualidade dos produtos e serviços fornecidos.

### 2.1.1 Gestão de qualidade de software

A gestão de qualidade de software se aplica a todas as perspectivas: de processos, produtos e recursos. Entre outros objetivos, estão envolvidos a definição de procedimentos, hierarquias, medidas e canais de *feedback* que permitam monitorar e melhorar a qualidade (37). A quantificação do desempenho de um processo de software bem como a qualidade do produto decorrente desse processo costuma ser avaliada por meio de métricas. A engenharia de software tem explorado cada vez mais formas de monitorar e controlar processos por meio da coleta de métricas (35) (11). Os chamados programas de métricas estabelecem todo um conjunto de práticas de coleta, análise e produção de indicadores voltados a informar a qualidade e desempenho de um PDS.

*A qualidade do produto está diretamente ligada com a qualidade do processo*. É comum ouvirmos essa frase. Tal expressão sintetiza o fato de que é difícil se obter um produto de alta qualidade derivado de um processo sem a formalização e controle adequados. A monitoração através de métricas é parte desse controle na engenharia de software. Seus indicadores permitem comparar numericamente objetivos com o estado atual do produto/processo durante seu desenvolvimento, de forma a manter um nível de qualidade e/ou desempenho dentro de padrões estabelecidos (exemplos: número máximo de defeitos x defeitos encontrados, custo estimado x custo real). No entanto, também se faz necessário monitorar a forma com que cada atividade é desenvolvida, buscando efetivamente garantir a qualidade do processo como um todo. Em PDS geralmente há profissionais dedicados a essa atividade. Voltado a qualidade de produtos, atividades de agentes ou equipes de SQA normalmente incluem a verificação sobre o cumprimento de requisitos de funcionalidade e

qualidade do software construído. Já com relação aos processos, o papel da área de SQA é garantir que todas as atividades e políticas organizacionais sejam cumpridas adequadamente, ou seja, de acordo com os padrões definidos pela organização.

Atualmente, empresas que buscam melhoria de qualidade tem se utilizado de modelos padronizados como SPICE (22), CMMI (34) (Capability Maturity Model Integration) e MPS.BR (49). Esses modelos servem como guia para a obtenção de processos gerenciados e otimizados para garantia de máxima qualidade e previsibilidade. A próxima seção trata com mais detalhes os modelos de maturidade adotados por empresas de software atualmente, dando enfoque ao modelo CMMI.

### 2.1.2 Modelos de qualidade de software

A competitividade e complexidade do mercado de software têm levado as empresas a voltarem cada vez mais as atenções a seus processos. Essas empresas buscam as melhores práticas de gerenciamento e desenvolvimento, geralmente baseadas em modelos de maturidade como aqueles citados na seção anterior. A adoção de tais padrões é útil tanto para as empresas, atuando na melhoria de seus processos, quanto para os clientes que podem assegurar-se de um nível de qualidade do produto/serviço que estão adquirindo. A seguir, é feita uma breve descrição do modelo CMMI. Trata-se de um conjunto de práticas para a formalização e melhoria de processos de engenharia. A literatura coloca o CMMI como um modelo de qualidade amplamente adotado por empresas de software atualmente (29) (24). Por este motivo, o modelo é adotado neste trabalho como referência na gestão de qualidade e melhoria de processos em PDS. Além de uma visão geral do modelo, são salientadas as áreas de maior interesse para este trabalho.

#### CMMI

O modelo CMMI (37) tem sido amplamente adotado como conjunto de práticas padrão para a garantia de qualidade e melhoria de PDS. Sua estrutura é baseada em níveis de maturidade, que por sua vez contêm as chamadas Áreas de Processo (AP) como elemento base. Cada AP engloba um conjunto de objetivos gerais e específicos para o gerenciamento e melhoria de processos e projetos de engenharia. O CMMI considera as áreas de processos sob duas perspectivas: de estágios e contínua. A representação por estágios utiliza conjuntos de AP predefinidas, que representam um caminho de melhoria para a organização. Cada conjunto de componentes define um nível de maturidade. A representação contínua permite que se selecione cada AP independentemente, realizando melhorias com um foco específico. O CMMI considera que toda a organização está, por definição, no nível 1 de maturidade. Este nível representa a ausência de controle formal dos processos e da qualidade de seus produtos.


Figura 2.1: Níveis de maturidade do modelo CMMI

Cada nível de maturidade CMMI é focado em um aspecto específico da organização. O nível 2 considera os processos como gerenciados a nível de projetos, onde cada projeto possui atividades de planejamento, monitoração, controle e avaliação de aderência à defi nição de seus próprios processos. O terceiro nível tem foco na definição e gerenciamento dos processos da organização. Através do chamado OSSP (Organization's Set of Standard Processes), devem ser estabelecidos todos os procedimentos, medidas e produtos de trabalho considerados padrão para o desenvolvimento de projetos da organização. O nível 4, gerenciado quantitativamente, sugere o controle dos processos através da análise de métricas, utilizando avaliação estatística ou outra técnica de análise quantitativa. O nível 5 caracteriza os processos como otimizados. É baseado no entendimento acerca das causas comuns de problemas e desvios do processo, buscando a melhoria contínua através da inovação e desenvolvimento organizacional.

Analisando as AP do CMMI separadamente, observa-se que diversos pontos salientam, direta ou indiretamente, a importância em monitorar os processos como forma de garantir eficiência e qualidade (37). Nesse aspecto, quatro AP são de particular interesse para este trabalho. São elas:

- Garantia de Qualidade do Processo e Produto

  Esta AP trata da avaliação de processos e produtos de trabalho. Seus objetivos gerais envolvem:

  - Avaliar objetivamente os processos realizados e produtos de trabalho contra os padrões e procedimentos aplicáveis;
  - Identificar e comunicar as inconformidades;
  - Garantir que todas as inconformidades identificadas sejam tratadas;

  - Prover *feedback* dos resultados das atividades de qualidade aos participantes dos projetos e à gerência;

- Foco no Processo da Organização

  O propósito dessa AP envolve planejar, desenvolver e implementar melhorias de processo baseadas em um profundo entendimento sobre os aspectos positivos e negativos dos processos organizacionais. Processos organizacionais incluem todos os processos adotados pela organização e seus projetos. Propriedades dos processos e possíveis pontos de melhoria são obtidas de várias fontes como medição, lições aprendidas, resultados de estimativas, *benchmarks* ou qualquer outra iniciativa de melhoria na organização.

- Gerenciamento Quantitativo de Projeto

  O gerenciamento quantitativo de projeto envolve várias questões relacionadas a processos, como:

  - Estabelecer e manter os objetivos de qualidade do projeto e desempenho dos processos;
  - Selecionar subprocessos, dentro dos processos definidos para serem estatisticamente gerenciados;
  - Estabelecer e manter o entendimento sobre as variações dos subprocessos selecionados utilizando medidas e técnicas estatísticas;

### 2.1.3 Considerações sobre a seção

Com base na análise do modelo CMMI, é possível concluir que iniciativas de medição e análise de aderência aos processos definidos são conceitos bastante presentes, seja em nível de projeto ou organizacional. Análise de desempenho de processos, detecção de desvios e suas causas devem também ser questões abordadas por empresas que buscam altos níveis de produtividade e qualidade de seus produtos. De nada adianta definir processos se não for possível garantir de alguma forma que as pessoas os sigam e que sejam eficientes, resultando em uma real contribuição para os objetivos da organização. Sob essa perspectiva, as AP citadas anteriormente constituem em motivadores para este trabalho. Outro fator de motivação da pesquisa, dentro da empresa parceira, está relacionada a uma iniciativa de renovação da certificação CMM-3 (para o atual CMMI-3) e possivelmente obtenção de níveis mais altos. O padrão CMMI foi apresentado aqui por dois motivos: (i) fazer parte do contexto da empresa parceira como modelo de qualidade; (ii) ser um modelo amplamente adotado na indústria de software. Entretanto, considera-se que os aspectos

discutidos acima devem receber igual importância em qualquer modelo de melhoria de processo.

Fica claro que modelos de melhoria de processos de software consideram importante a análise de aderência entre processos definidos e executados. Porém, uma questão importante a considerar é que tais modelos costumam descrever o *que* deve ser feito, e não *como* fazer. A forma com que os objetivos apontados serão alcançados fica a cargo de cada organização. Logo, qualquer empresa engajada em uma iniciativa de melhoria deve encontrar formas efetivas de monitorar seus processos. O presente trabalho envolve monitoração de processos com o objetivo de verificar conformidade entre o processo formalizado e sua execução. Técnicas de extração automática de conhecimento sobre dados de execução (mineração de processos) são exploradas como uma ferramenta interessante visando, não só facilitar a tarefa de verificação, como aumentar a objetividade dos resultados providos pelos métodos tradicionais de auditoria.

## 2.2 Conformidade de processos

### 2.2.1 Definição

Segundo a Workflow Management Coalition[1](13) um processo de negócio, ou simplesmente um processo, é *"um conjunto de tarefas interligadas, desenvolvidas por um grupo de recursos dentro de uma estrutura organizacional, com um determinado objetivo de negócio"*. A fim de padronizar procedimentos e obter melhor controle, processos de negócio costumam ser formalizados de acordo com alguma notação. Atualmente é possível encontrar uma grande variedade de modelos de processos, em todo o tipo de organização e com os mais diversos propósitos. Esses modelos podem ser simples fluxogramas ou descrições formais complexas de workflow (utilizadas como referência para a execução de sistemas de gerencia de workflow como Oracle Workflow, FLOWer, Staffware, etc.). Eles descrevem como um determinado processo de negócio deve ser executado, a sequência de atividades, papéis envolvidos e regras de negócio necessárias para o cumprimento dos objetivos do processo. No entanto, modelos de processos não produzem resultados sozinhos. Normalmente processos de negócio envolvem pessoas, em ambientes dinâmicos, onde nem sempre é possível seguir todas as regras estabelecidas em um modelo formal. Alguma distância entre o formal e a realidade sempre existe. Análise de conformidade de processos procura determinar o grau de semelhança entre o processo que é realmente executado e aquele que se espera que o seja.

[1]Entidade sem fins lucrativos, criada em 1993 com o objetivo de estabelecer padrões e terminologias internacionais na área de workflow[2] e processos de negócio

### 2.2.2 Motivação

Existem diversas razões que tornam a análise de conformidade um tópico importante (46). A confiança em um modelo de processo formal, por exemplo, é elevada quando se pode mostrar que a execução efetiva do processo é consistente com o modelo prescrito. Por outro lado, as informações obtidas podem levantar necessidades de evolução no processo para acomodar novas regras e atividades. De forma geral, busca-se a aderência de processos para:

- Garantir uma execução de processo estável e previsível;
- Garantir a validade dos dados, informações, experiências e conhecimento adquiridos durante o processo;
- Garantir que a execução do processo satisfaça determinados requisitos como por exemplo, um modelo de certificação;
- Reduzir tempos e custos, na medida em que se possui um maior conhecimento e controle sobre os processos;

### 2.2.3 Conformidade de processos na Engenharia de Software

Sempre que o modelo de um processo é criado, a questão que surge é se aquele modelo captura com fidelidade o ambiente. Na área de sistemas de software, onde o modelo é tipicamente integrado e executado com auxílio de diversas ferramentas de software, essa questão é evitada. O modelo e o processo são supostos necessariamente aderentes porque o modelo se torna o processo (17). Porém, quando aplicada à engenharia de software na prática essa consideração nem sempre é verdadeira. Em particular, se assume que todo o processo é executado dentro do ambiente automatizado. Na verdade, aspectos críticos como tomadas de decisão e comunicação entre os participantes ocorrem fora do computador e, dessa forma, não estão sob o controle desses sistemas. Sendo assim, não há uma forma efetiva de forçar a aderência ao processo, não sendo possível garantir a consistência entre o modelo formal e sua real execução. Mesmo que exista uma maneira de garantir a tal aderência, ainda existe o problema em gerenciar mudanças nos processos. Desvios acontecem e devem, dentro de certo limite, serem suportados. Dessa forma, faz-se necessário obter métodos para detectar e caracterizar as diferenças entre o modelo formal e execução efetiva de processos.

Conformidade de processos tem recebido diferentes nomes em diferentes contextos. Além da expressão *conformance*, definições semelhantes também são conhecidas como *correctness* (23), *fidelity* (31) e *compliance*. Fagan (23) foi um dos primeiros autores a reconhecer a importância do tema relacionado à engenharia de software, ao desenvolver

técnicas de inspeção (23). O mesmo autor ressalta a importância de se executar processos de forma completa e correta, sendo derivado daí o termo *correctness* (23) nesse contexto. O tema também está diretamente ligado à área de Software Process Improvement (SPI). Em conjuntos de práticas de melhoria como o CMMI, desvios dos processos definidos são considerados uma ameaça ao desempenho. A menos que esses desvios sejam rastreados, revisados e registrados, eles irão acarretar degradação e descontrole sobre os processos com consequente perda de qualidade (31). Na literatura são identificadas três principais maneiras de verificar conformidade em processos de software (46):

- A realização de entrevistas é uma forma. É um método simples e flexível, não exigindo ferramentas ou modelos formais. Podem ser aplicadas a qualquer processo. Porém, é mais difícil obter informações quantitativas dos resultados. Além disso, a confiança nas respostas pode ser questionável. Pode ser difícil para a pessoa entrevistada fornecer as informações necessárias sobre a execução do processo. Outro problema com entrevistas é que elas exigem tempo e trabalho intensos.

- A segunda abordagem é baseada na execução do modelo formal do processo suportada por um sistema computacional. Na verdade nesse caso a conformidade pode ser garantida, mais do que medida. Isto porque o processo é executado sob o comando do sistema computacional. Mesmo provendo meios de medir a conformidade, nem sempre essa informação é disponível diretamente. Partes do processo que ocorrem fora do computador também se tornam difíceis de monitorar. A diminuição na flexibilidade do processo também é uma questão a ser considerada. Isto porque a execução do processo se torna restrita ao fluxo de execução determinado pelo sistema.

- Uma terceira forma de monitorar processos utiliza algoritmos aplicados a fluxos de eventos do modelo formal e sua execução para comparar as diferenças entre os dois. Embora originalmente voltado à descoberta de modelos de processo, o termo *mineração de processos* também tem sido relacionado à verificação de conformidade utilizando essa abordagem. O principal ponto em comum é a extração de padrões recorrentes a partir de um conjunto de eventos, provendo informações sobre a realidade de execução de um processo.

## 2.3 Mineração de processos

### 2.3.1 Definição

Mineração de processos surgiu como uma forma de analisar sistemas e seu uso efetivo, baseado no log de eventos produzidos por esses sistemas. O assunto tem sido amplamente pesquisado nos últimos anos (4) (7) (43) (12). O tema possui estreita relação com

o conceito clássico de mineração de dados. Segundo Han et al. (28), mineração de dados refere-se à *"extração de conhecimento de grandes quantidades de dados, por meio automático ou semi-automático, a fim de descobrir padrões e regras significativos"*. De forma semelhante, a mineração de processos concentra-se na extração de padrões. Porém, ela é voltada para as relações de sequência e concorrência entre eventos. Enquanto a mineração de dados concentra-se em evidenciar tendências nos dados e relacionamentos entre atributos, com a mineração de processos busca-se entender como um determinado processo é executado, com base na análise de cada uma de suas instâncias. Perguntas comuns sobre os processos podem ser:

- Qual o caminho mais frequente no processo?
- Quais são as estruturas de roteamento (E/OU) para tarefas concorrentes?
- As regras de negócio no processo são realmente obedecidas?
- Onde estão os gargalos do processo?
- Quais as pessoas envolvidas em determinada atividade?
- Qual o fluxo de comunicação entre as pessoas/sistemas?

A mineração de processos sempre inicia com um log de eventos. Tais registros são produzidos por praticamente qualquer sistema de informação (desde sistemas bancários e gerenciadores de workflow até redes de sensores e *web services*). Na maioria dos casos, um log registra o início e/ou conclusão de eventos além do momento de execução (*timestamp*). Em alguns sistemas, o responsável pela execução da atividade e outros dados adicionais também são registrados. São encontradas na bibliografia inúmeras soluções de descoberta de modelos de processos, que têm seguido basicamente duas frentes: (i) mineração de interação entre web services (20) e (ii) mineração de workflow (*software process discovery* ou simplesmente *process discovery*) (4) (30) (38). Na mineração de interação de *web services* o objetivo é inferir um modelo de processos que represente a interação entre diversos serviços. O esforço da mineração de workflow concentra-se em descobrir um workflow capaz de modelar o fluxo de controle dentro de um determinado processo. A principal diferença entre essas duas abordagens é que, enquanto a primeira analisa mensagens trocadas entre serviços que modificam o estado do sistema, a segunda utiliza o conceito de eventos/atividades para a inferência de modelos de workflow.

A maioria das organizações documenta seus processos de alguma forma. As razões para tal são diversas. Modelos de processo podem ser utilizados para comunicação, configuração de sistemas, análise, simulação, entre outros. Adicionalmente, um modelo de processo pode ser de natureza descritiva ou prescritiva (42). Modelos descritivos tentam

capturar o ambiente de processo existente de forma não restritiva. Como exemplo, em um procedimento hospitalar deve ser possível reagir a situações de urgência. Entretanto, a flexibilidade para divergir do fluxo normal do processo definido é fundamental. Modelos prescritivos descrevem de forma mais rígida a maneira que os processos devem ser executados. Em sistemas de gerenciamento de workflow, modelos formais são utilizados para imprimir um fluxo específico de execução do processo de negócio.

Em qualquer um dos casos descritos acima, a descoberta de conhecimento sobre processos pode ser de grande utilidade. No caso de já se existir um modelo de processo predefinido em um contexto menos restritivo, surge a necessidade de verificar se o que foi planejado é realmente seguido. Por outro lado, seja de maneira descritiva ou prescritiva, uma fase de modelagem do processo de negócio é sempre necessária. Diversos autores afirmam que essa etapa normalmente consome muito tempo, trabalho, sendo ainda sujeita a erros (4) (12) (30). Nesse contexto, a mineração de processos pode ser aplicada buscando facilitar a obtenção do modelo de processo com base na exploração de eventos registrados durante a execução.

### 2.3.2 Propósitos e perspectivas

O propósito da mineração de processos é descobrir, monitorar e melhorar processos extraindo conhecimento de eventos produzidos. A forma do conhecimento extraído está tipicamente relacionada aos objetivos que se tem com a aplicação da mineração de processos. Autores como Rubin (43) e Aalst (4) estabelecem duas classificações básicas: (i) perspectiva; (ii) propósito. Conforme os dados explorados, o processo pode ser analisado sob basicamente três perspectivas. Na perspectiva conhecida como de *fluxo de controle*, o objetivo é analisar as relações de dependência entre os diversos eventos/atividades do processo. Notações comumente utilizadas para representação são Redes de Petri ou FSM (*Finite State Machines*) (12). Uma segunda visão está relacionada ao entendimento sobre as relações de cooperação entre os envolvidos no processo (pessoas/sistemas). Essa perspectiva é conhecida como *organizacional*. Há ainda uma terceira visão, chamada de *informação*. Aqui, o objetivo é compreender o conteúdo dos dados envolvidos em cada instância (ex.: em um processo de venda, uma atividade *fatura* com valor acima de 10000 exige que a atividade *confirmar_fatura* seja executada duas vezes). De modo geral, essas três perspectivas buscam responder as questões, respectivamente "*Como?, Quem?* e *O quê?*".

Além da visão sob as perspectivas descritas acima, a mineração de processos pode ser realizada com os seguintes propósitos:

- Descoberta: Não existe um modelo formal do processo definido *a priori*. O modelo pode ser extraído dos próprios dados produzidos durante a execução do processo.

- Análise de conformidade: Neste caso conta-se com algum tipo de modelagem formal do processo. No entanto, o que se busca é descobrir se a realidade está conforme com o modelo. A análise de conformidade pode ser utilizada para detectar desvios do processo, auxiliar a entender suas causas e medir a severidade de tais desvios.
- Extensão: O objetivo aqui é enriquecer, melhorar um modelo pré-existente a partir da análise de dados de execução. A análise de dados de desempenho de um processo, por exemplo, poderia ser utilizada para eliminar gargalos e otimizar o processo.

A figura 2.2 fornece uma visão geral acerca das aplicações de mineração de processos em um ambiente suportado por algum tipo de sistema de informação.

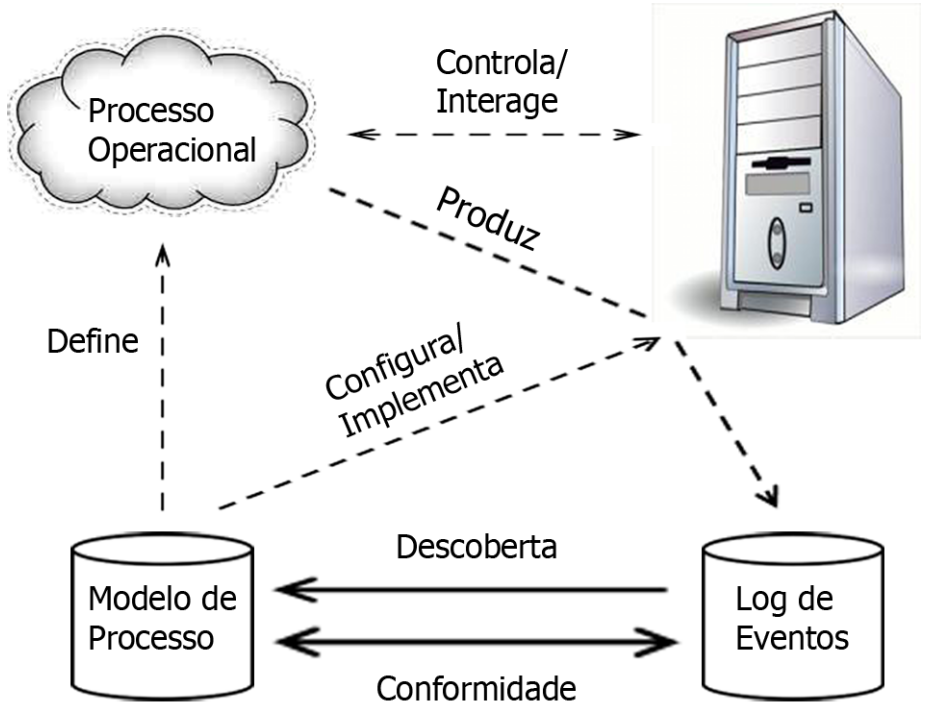


Figura 2.2: Visão geral de um ambiente de mineração de processos.

### 2.3.3 Redes de Petri

Em certos casos é necessário modelar o comportamento dinâmico de um sistema ou processo. Por comportamento dinâmico entende-se a representação de um fluxo completo de execução, expressando cada estado pelo qual o sistema/processo passa de acordo um conjunto de regras (condições). Uma notação amplamente utilizada para tal representação é a rede de Petri (1). Seu uso se torna interessante porque, além de ser uma notação gráfica de fácil entendimento, permite a aplicação de diversas técnicas formais de verificação e simulação. Trata-se de uma estrutura dinâmica, modelada graficamente, que consiste em

um conjunto de *transições*, *lugares* e *arcos*. As *transições*, indicadas por retângulos, são relacionados a alguma atividade ou ação que pode ser executada no sistema. Entre transições existem *lugares*, que podem conter *tokens*. A distribuição de *tokens* representa um determinado estado na rede. Por fim, *transições* e *lugares* são ligados por arcos direcionados. Além de conexões simples, esses arcos podem representar situações de concorrência e escolha (AND-Split, OR-Split, AND-Join, OR-Join). Os tipos de comportamento representados são conhecidos como padrões de fluxo. Para uma transição ocorrer ela deve estar habilitada. Transições estão habilitadas no momento em que todos os seus lugares de entrada contém um *token*. Estando habilitada a transição pode ser executada, consumindo um *token* de cada uma de suas entradas e produzindo um *token* para cada uma de suas saídas. Abaixo são listados os cinco padrões de fluxo básicos para redes de Petri.

- *Sequential routing* é a ligação direta entre duas transições, sendo caracterizado pelo fato de uma transição ter início apenas após o término da execução da transição anterior.
- *AND-Split* consiste em um ponto na rede onde um único fluxo se divide em múltiplos fluxos do controle que podem ser executadas paralelamente.
- *AND-Join* é um ponto onde diferentes fluxos paralelos convergem em um único fluxo de controle. Para que a execução da transição fim dos fluxos paralelos prossiga, é necessário que todos esses lugares contenham *tokens*, estando a transição dessa forma habilitada.
- *OR-Split* caracteriza-se pela divisão do fluxo, baseada em condições de decisão ou controle que determinam um único caminho a seguir.
- *OR-Join* é um ponto na rede onde dois ou mais fluxos de execução unem-se sem que haja uma sincronização, ou seja, não são transições concorrentes.

### 2.3.4 Um exemplo

Para que um algoritmo de mineração de processos possa inferir um modelo a partir de um log, são assumidas algumas premissas. São elas: (i) cada evento se refere a uma atividade no processo, (ii) um evento pertence a uma instância específica e iii) eventos estão totalmente ordenados ou possuem um timestamp indicando o momento de sua execução. Para o caso de eventos não atômicos (execução não instantânea), o estado da atividade também deve estar presente (iniciada, completa, suspensa, etc.). Partindo dessas premissas básicas já se torna possível descobrir a relação entre os eventos contidos no log analisado. Adicionalmente, um log pode conter o responsável pela execução de cada atividade. Esse dado possibilita extrair informações sob a perspectiva organizacional citada na

seção 2.3.2. Para ilustrar a aplicação da mineração de processos, sob as perspectivas de fluxo de controle e organizacional, considere a figura 2.3. Ela mostra um exemplo de log envolvendo 19 eventos, 5 atividades e 6 atores. O log contém 5 instâncias de processos. É possível observar que em quatro delas (1, 2, 3 e 4) as atividades A, B, C e D foram executadas. Na quinta instância apenas três foram executadas: A, E e D. Toda instância começa com a execução de A e termina executando a atividade D. Se a atividade B é executada, C também é. Entretanto, em algumas instâncias a atividade C é executada antes de B. Baseado nessa informação, um algoritmo de mineração poderia encontrar o modelo da figura 2.4. Um ponto importante a ressaltar aqui é que, nesse caso, assume-se que o número de instâncias é suficiente para representar o ambiente observado. O modelo está em notação de redes de Petri (1), contendo informações sobre as perspectivas de fluxo de controle e organizacional. Além da relação de sequência entre as atividades, o modelo mostra o envolvimento dos atores com uma das atividades. À esquerda da figura os atores envolvidos em uma mesma atividade estão agrupados por papéis (ex.: *Role* X para a atividade A). À direita é apresentado um diagrama conhecido como *sociograma*, ou seja, as relações de transferência de trabalho entre os colaboradores envolvidos no processo.

| case id | activity id | originator | timestamp |
|---|---|---|---|
| case 1 | activity A | John | 9-3-2004:15.01 |
| case 2 | activity A | John | 9-3-2004:15.12 |
| case 3 | activity A | Sue | 9-3-2004:16.03 |
| case 3 | activity B | Carol | 9-3-2004:16.07 |
| case 1 | activity B | Mike | 9-3-2004:18.25 |
| case 1 | activity C | John | 10-3-2004:9.23 |
| case 2 | activity C | Mike | 10-3-2004:10.34 |
| case 4 | activity A | Sue | 10-3-2004:10.35 |
| case 2 | activity B | John | 10-3-2004:12.34 |
| case 2 | activity D | Pete | 10-3-2004:12.50 |
| case 5 | activity A | Sue | 10-3-2004:13.05 |
| case 4 | activity C | Carol | 11-3-2004:10.12 |
| case 1 | activity D | Pete | 11-3-2004:10.14 |
| case 3 | activity C | Sue | 11-3-2004:10.44 |
| case 3 | activity D | Pete | 11-3-2004:11.03 |
| case 4 | activity B | Sue | 11-3-2004:11.18 |
| case 5 | activity E | Clare | 11-3-2004:12.22 |
| case 5 | activity D | Clare | 11-3-2004:14.34 |
| case 4 | activity D | Pete | 11-3-2004:15.56 |

Figura 2.3: Exemplo de um log de eventos. Fonte: (2)

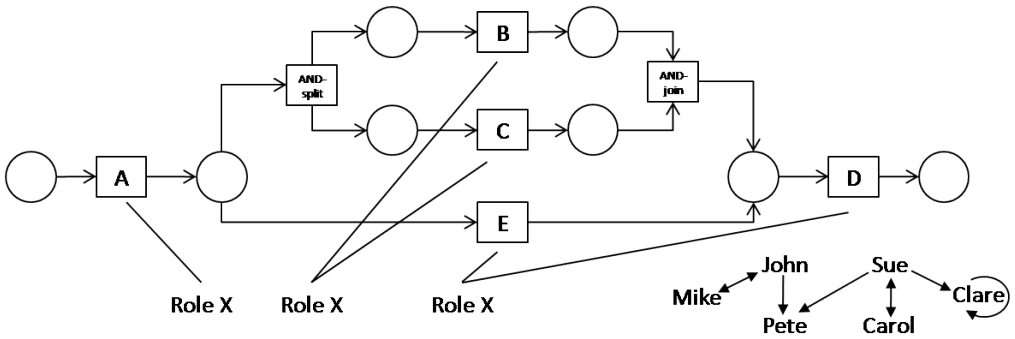


Figura 2.4: Possível modelo representando o log da figura 2.3. Fonte: (2)

### 2.3.5 *Delta Analysis* e *Conformance Testing*

Analisar conformidade está entre as várias aplicações possíveis da mineração de processos. De um lado se tem um modelo predefinido do processo. De outro, obtendo registros de eventos de execução desse mesmo processo, é possível comparar o grau de similaridade entre essas duas visões. Na área de métodos formais, diversas noções de equivalência já foram propostas (19) (25). Entretanto, a maioria dessas abordagens fornece respostas do tipo ”verdadeiro/falso”. Tais respostas não são muito úteis na comparação de modelos de processos de negócio, já que raramente existirá uma equivalência perfeita. Torna-se mais interessante obter um determinado grau de similaridade, expressado de forma quantificável. Uma das formas conhecidas para analisar conformidade por mineração de processos é a chamada *delta analysis*. Nessa abordagem, um modelo descoberto por meio de mineração é comparado com o modelo predefinido. A verificação pode se dar, tanto de forma visual, quanto a partir da aplicação de técnicas formais de comparação de modelos com a obtenção de métricas de conformidade (39). Para ilustrar a aplicação de *delta analysis*, é apresentado a seguir um exemplo retirado de (2). A figura 2.5 ilustra o exemplo de um processo de compras, representado em notação *Workflow-Net* (Wf-Nets) (1), que é uma subclasse da notação clássica de redes de Petri. Os quadrados representam as atividades enquanto os círculos correspondem à parte passiva do processo, representando estados. A figura 2.5 apresenta o modelo prescritivo do processo. Essa seria a forma como o processo é ”percebido” pelos gestores. Agora suponha que por meio do log[3] da figura 2.7 fosse possível descobrir o modelo da figura 2.6. Ao comparar os dois modelos, quatro diferenças básicas podem ser observadas: 1) no processo descoberto, em caso de falta de estoque uma ordem de renovação é sempre emitida (*replenishment*, 2) as atividades *send_bill* e *receive_payment* estão em paralelo no modelo descoberto. Isso sugere que talvez em algumas instâncias, clientes pagam antes de receber a fatura, 3) no modelo descoberto não é possível enviar múltiplas faturas e 4) no modelo descoberto é possível emitir a fatura antes

[3]Para fins de ilustração, é mostrado apenas o fragmento de um log. Em um caso real, um volume muito maior de eventos seria necessário para inferir um modelo

de despachar os produtos.

Cada uma dessas diferenças fornece pistas interessantes sobre a discrepância entre o processo real observado e o que se espera que seja realizado. Se um sistema de informação é configurado conforme a figura 2.5 e as pessoas na realidade trabalham como sugerido pelo modelo descoberto (figura 2.6), existe uma inconformidade. Esse problema poderia ser resolvido atualizando o modelo definido ou incentivando as pessoas a aderirem ao processo original. Nesse exemplo simples, contudo, é fácil identificar visualmente as diferenças entre os modelos. Porém, para processos complexos isso pode não ser possível. Existem maneiras de destacar graficamente as discrepâncias entre modelos desse tipo. Aalst destaca duas abordagens mais relevantes: *inheritance of behavior* (3) e *change regions* (21). A primeira utiliza noções de herança do desenvolvimento orientado a objetos aplicadas a processos. Resumidamente, são buscadas sub-classes ou super-classes em comum nos dois modelos. A segunda abordagem provém da área de adaptação dinâmica de workflows. A estratégia é identificar, por artifícios matemáticos, regiões no modelo original que são diretamente afetadas, quando comparadas ao modelo modificado. Essas regões são chamadas *change regions*.

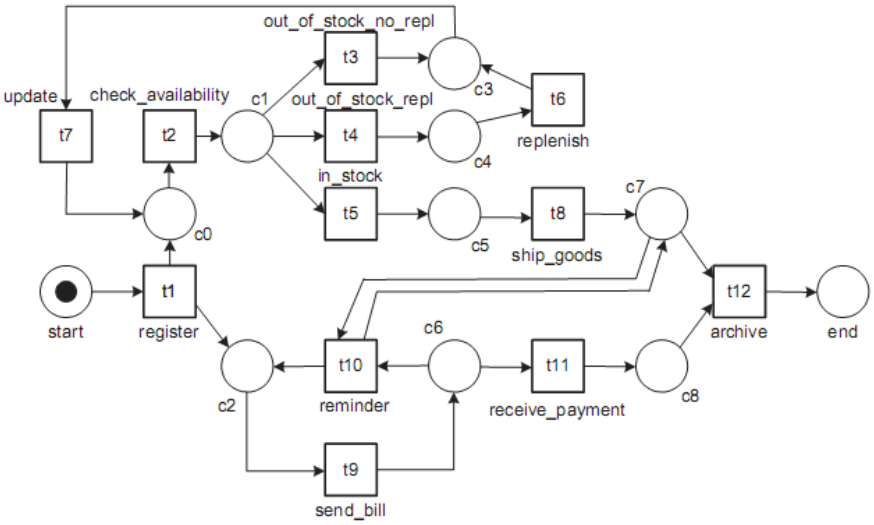


Figura 2.5: Modelo predefinido do processo. Fonte: (2)

No método de *delta analysis* são necessários para a comparação, um modelo descritivo/prescritivo do processo e um modelo descoberto por mineração. Porém Cook e Wolf (17) e mais tarde Rozinat e Aalst (2) propõem a abordagem *conformance testing*, que não necessita dos dois modelos. A comparação é feita diretamente entre o modelo predefinido e o log de execução, visto que nem sempre se tem dados suficientes para descobrir um modelo comparável com o predefinido. Dado um modelo e um conjunto de instâncias do processo, é possível verificar se cada uma dessas instâncias é compatível com o modelo ou não. No caso de um modelo em notação de redes de Petri, por exemplo, cada evento

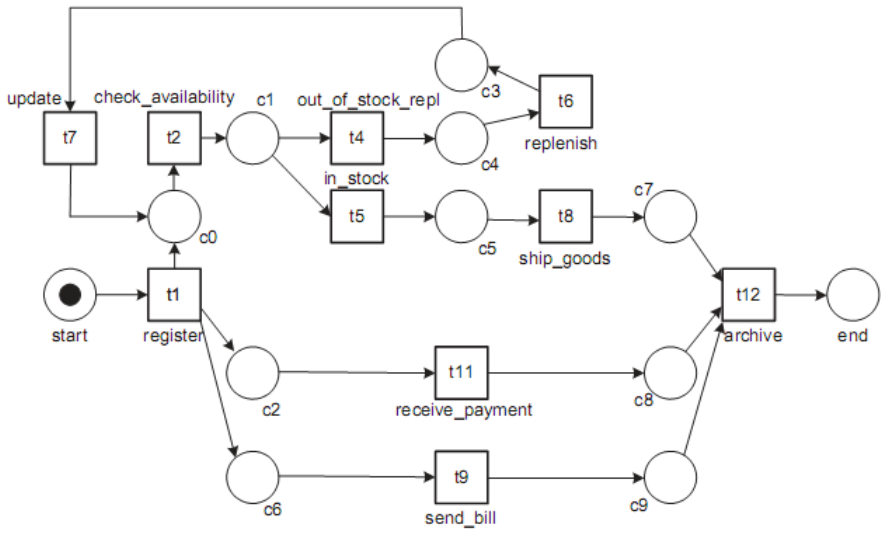


Figura 2.6: Modelo descoberto por mineração de processos. Fonte: (2)

| case id | activity id | originator | timestamp |
|---|---|---|---|
| case 1 | register | Pete | 14-3-2004:14.33 |
| case 2 | register | Carol | 14-3-2004:15.22 |
| case 3 | register | Chris | 14-3-2004:16.43 |
| case 3 | send bill | Michael | 15-3-2004:16.17 |
| case 1 | receive payment | Jorge | 15-3-2004:18.05 |
| case 1 | check availability | Pete | 16-3-2004:9.43 |
| case 2 | send bill | Mike | 16-3-2004:10.24 |
| case 4 | register | Sue | 16-3-2004:10.35 |
| ... | ... | ... | ... |

Figura 2.7: Fragmento de um log de eventos utilizado para descobrir o processo. Fonte: (2)

no log dispara uma transição. Dessa forma, é possível verificar se a sequência de eventos contida na instância do processo é executável no modelo predefinido. A técnica de *Conformance Testing* também pode fornecer métricas sobre o nível de conformidade entre o modelo e o log. Métricas essas que podem ser obtidas com base no número de eventos, de atividades e as instâncias que podem ser executadas com sucesso no modelo. Considere *P* como um modelo de processo e *L* um log. Uma instância *c* em *L* é executada com sucesso se, e somente se, *c* corresponder a uma sequência possível em *P*, sendo que após sua execução existe um *token* no estado final. No entanto, o método de *conformance testing* se interessa em descobrir quando o modelo não corresponde à execução e onde as inconformidades acontecem, expressando as diferenças quantitativamente. Nos pontos onde há discrepâncias, as transições disparadas a partir do log criarão *tokens* em excesso ou haverá *tokens* faltando para executar a transição corretamente no modelo. Entre outros dados, os algoritmos utilizam a relação entre *tokens* faltantes e restantes para obter as métricas de conformidade. As métricas propostas em (17) e em (42) são explicadas em mais detalhes no capítulo 4.

## 2.4 Considerações sobre o capítulo

Nesse capítulo foram discutidos aspectos relacionados à práticas de gestão de conformidade de processos como ferramenta para obtenção de alta qualidade e produtividade na engenharia de software. Procurou-se, com isso, esclarecer como tais práticas se enquadram no contexto de PDS, destacando a importância e os desafios na gestão de conformidade de processos. O conceito de mineração de processos foi apresentado como método inovador, que desperta grande interesse atualmente tanto no auxílio à descoberta de modelos de processos quanto na verificação dos mesmos. Fazendo ligação com o ambiente de PDS o trabalho aborda, de forma ampla, como práticas de análise de conformidade com mineração de processos podem ser aplicadas baseadas na documentação e estrutura computacional típicas de uma empresa de software.

# 3. Cenário

## 3.1 Descrição

A pesquisa foi desenvolvida em colaboração com a empresa HP (*Hewlett-Packard Company Brasil Ltda*), mais precisamente, em sua operação de software instalada no TECNOPUC - Parque Tecnológico da PUCRS em Porto Alegre. A operação está em funcionamento a cerca de cinco anos, desenvolvendo grandes projetos de software para instituições públicas e privadas. Em novembro de 2005, a empresa obteve a certificação de nível 3 CMM, sendo parte deste esforço realizado em parceria com a PUCRS. O trabalho de certificação de fato trouxe resultados positivos para a melhoria de qualidade e produtividade da empresa. A formalização e documentação de todo o conjunto de processos adotados é um dos produtos resultantes do trabalho. Trata-se inclusive de requisito essencial para qualquer organização certificada em nível 3. Outro aspecto importante foi a implantação de um repositório central de métricas para todos os projetos da organização, chamado de Base Organizacional (BO). O projeto foi desenvolvido em uma parceria entre o Programa de Pós-Graduação de Ciência da Computação da PUCRS (PPGCC-PUCRS) e a HP, para apoiar o programa de métricas da empresa. O propósito de tal repositório é prover uma visão consistente da realidade dos projetos, visando auxiliar os gestores no processo de tomada de decisão.

Na área de garantia de qualidade, uma equipe de SQA é encarregada de garantir o controle de qualidade dos produtos desenvolvidos, além de monitorar a aderência dos colaboradores aos processos definidos. Note-se que este também é um requisito do nível 3 do modelo de maturidade. Aderente ao padrão CMM, a empresa possui um conjunto de processos organizacionais (OSSP). Além disso, cada projeto pode realizar adaptações nesses processos de acordo com necessidades específicas. As adaptações e dispensas, depois de validadas e autorizadas pela gerências sênior e equipe de SQA, são documentadas no PDSP (Project's Defined Standard Processes). A equipe de SQA realiza periodicamente, auditorias de produto e processo com base nesses documentos e nas evidências coletadas dos projetos. O objetivo é manter visibilidade sobre a qualidade dos produtos desenvolvidos e dos processos utilizados para tal. Cabe salientar aqui que, embora a estrutura dos processos da empresa esteja baseada no modelo CMM, tal estrutura é muito semelhante à encontrada no padrão atual CMMI (37). O modelo CMMI possui apenas algumas diferenças conceituais e de nomenclatura, já que trata de processos de engenharia de forma geral e não mais de processos de software como o antigo CMM. Sendo assim, assume-se que as características apresentadas de acordo com o CMM sejam perfeitamente aplicáveis ao atual CMMI.

## 3.2 Suporte Computacional

Como suporte computacional na gestão dos projetos a empresa conta com diferentes ferramentas, sendo cada uma voltada a um aspecto de projeto específico. A função das principais ferramentas são: (i) gestão de cronograma, (ii) controle de demandas e defeitos para projetos de manutenção, (iii) gestão de requisitos, (iv) gerência de configuração, (v) gestão de testes e (vi) coleta de métricas. Considera-se que esse tipo de ambiente seja comumente encontrado em empresas de software de médio e grande porte. Analisadas pela visão de conformidade de processos, duas questões interessantes são observadas aqui. Por um lado, esse conjunto de ferramentas é capaz de produzir um grande volume de registros históricos de execução, podendo servir de fonte para a monitoração de processos. Mas, por outro lado, surge a dificuldade em identificar, extrair e estruturar todos esses dados para produzir informações relevantes sobre o processo executado.

## 3.3 Base Organizacional

A empresa possui um repositório de métricas para o controle de indicadores de seus projetos. A concepção da BO, teve como meta principal prover uma visão unificada dos diferentes projetos da operação, de forma a viabilizar a comparação e análise dos mesmos de maneira sistemática e confiável. O projeto envolveu a parceria PUCRS/HP. Mais detalhes sobre o trabalho podem ser encontrados em (11). Em um ambiente de *Data Warehousing*, são integrados e consolidados os dados provenientes de todo o ambiente computacional utilizado nos projetos. Mensalmente, é realizado um processo de carga de dados nessa base, para torná-los acessíveis à gerência por meio de uma interface de BI (*Business Intelligence*). Esse processo pode ser dividido em cinco etapas. A figura 3.1 ilustra o fluxo de carga das métricas. Primeiro, os colaboradores atuantes nos respectivos projetos (representado na Figura 3.1 pelos paralelogramos) obtêm as métricas no PDS, ilustrado como a primeira etapa na figura. Como segunda etapa ocorre o registro dessas métricas nas diferentes ferramentas que o projeto tem para apoio no gerenciamento. Na terceira etapa ocorre a construção de pacotes onde são agrupados os dados mensais, proporcionando uma visão do projeto naquele período. A quarta etapa é ilustrada pelo armazenamento dos pacotes no repositório central para, na quinta e última etapa, serem extraídos e apresentados aos gerentes e gestores na interface de BI.

## 3.4 Auditorias de SQA

Para verificar aderência dos colaboradores aos processos definidos uma equipe de SQA realiza, normalmente uma vez ao mês, auditorias em todos os projetos da empresa. O principal artefato utilizado é uma planilha de *checklist* desenvolvida internamente, de forma

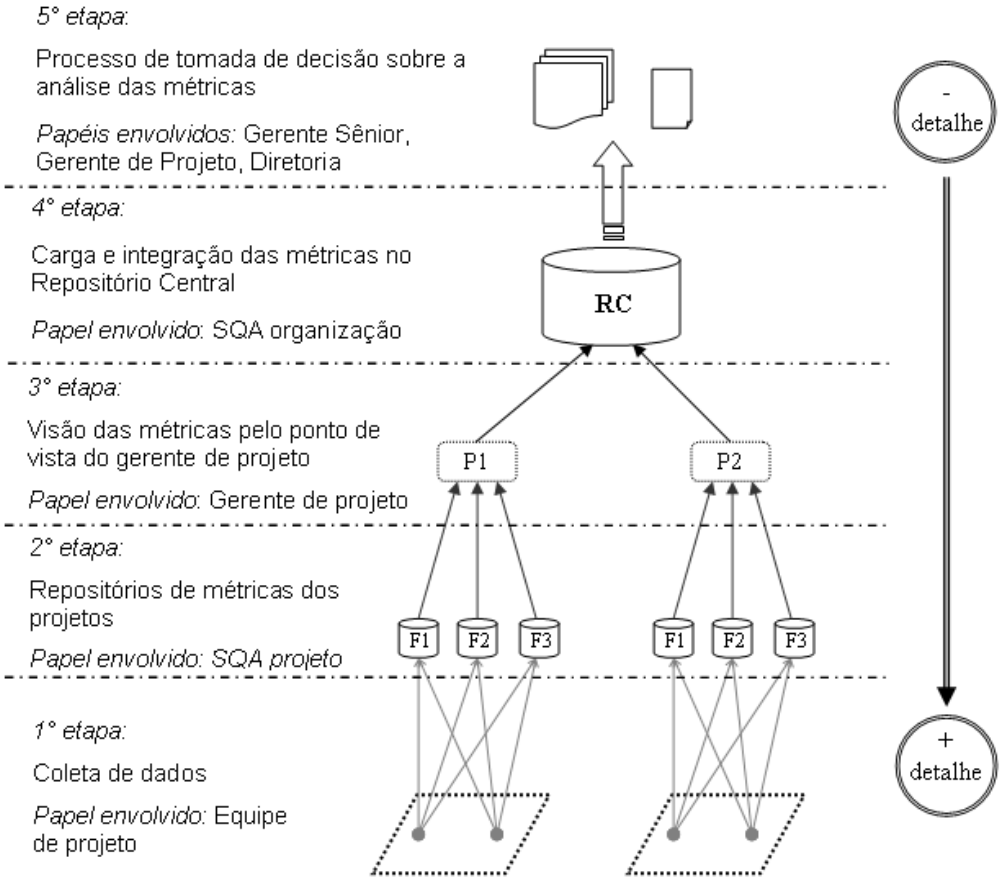


Figura 3.1: Fluxo de registro das métricas organizacionais

aderente ao modelo CMM-3. Essa planilha contempla todas as fases de projeto, contendo cada uma delas uma série de itens de verificação. Cada uma dessas questões diz respeito a requisitos relacionados, tanto aos produtos (documentos, código ou qualquer outro artefato produzido), quanto às responsabilidades de cada colaborador dentro do projeto. Tendo o *checklist* como base, a equipe realiza entrevistas e analisa documentos buscando evidências acerca da aderência dos seus colaboradores aos processos organizacionais. Então os resultados são compilados e reportados à gerência. A partir daí, são discutidas iniciativas de resolução dos desvios encontrados e estipulados prazos para o fechamento desses desvios.

## 3.5 Caracterização do problema

A parceria proporcionou a interação direta com a equipe de qualidade da empresa. Essa experiência mostrou, contudo, que analisar conformidade de processos em uma grande empresa de software não é uma tarefa simples. Os resultados dependem de muitos aspectos subjetivos, como: habilidade de comunicação do auditor, disponibilidade dos colaboradores para o fornecimento de informações e formato das questões utilizadas nas auditorias. Além

disso, a presença do auditor pode interferir no comportamento natural do colaborador. Esse fenômeno é inclusive bastante conhecido, chamado de efeito *Hawthorne* (45). Observou-se também que, embora bem documentados, os processos não são impostos ao colaboradores. Eles ficam acessíveis ao time, incluindo gráficos, guias e *templates*. Entretanto, não se pode garantir que cada membro da equipe realmente acesse frequentemente a documentação, seguindo assim o processo fielmente. Sabe-se que a implantação de formas de gestão mais rígidas, baseados em sistemas de automação de workflow, pode restringir tal subjetividade. Entretanto, esse tipo de controle possui dois problemas: (i) Definir um modelo *a priori* para um processo complexo como um PDS é trabalhoso, exige muito tempo e constantes atualizações a cada mudança de comportamento no processo ou situações que não foram identificadas previamente, (ii) PDS são bastante dinâmicos, necessitando de certa flexibilidade, o que pode ser fortemente restringido pela gestão de um sistema de workflow. Dessa forma, a maior dificuldade está justamente em estabelecer um ponto de equilíbrio que permita manter o dinamismo característico de um PDS, obtendo ao mesmo tempo a visibilidade e o controle adequados para garantir a qualidade e o desempenho desejado nos processos.

Tendo em vista os fatores discutidos acima, surge a motivação de buscar métodos que possam complementar e facilitar o processo de análise de conformidade no ambiente de PDS. Conforme apresentado na seção 2.3, técnicas automáticas de descoberta de conhecimento sobre processos tem sido aplicadas nas mais diversas áreas. Além da descoberta de modelos, diversas aplicações existem também com o objetivo de analisar conformidade em processos (6) (42) (17). Então, o objetivo do trabalho concentra-se em explorar as possibilidades e resultados potenciais da aplicação de uma solução mineração de processos na análise de conformidade em um ambiente de PDS. É importante destacar que, embora baseado no contexto da empresa parceira, o objetivo busca abordar o tema de forma genérica. Levamos em conta o fato de a empresa ter sido certificada CMM-3, tendo o seu processo de SQA avaliado e aprovado. Com isso, assume-se que o tipo de método de análise de conformidade encontrado na empresa parceira seja também adotado predominantemente em empresas de software baseadas nos mesmos padrões.

### 3.5.1 Requisitos da solução

A visão de uma solução ideal para verificação de conformidade de processos em PDS é proveniente de duas fontes. Primeiro, devido à participação direta no processo de auditorias juntamente com o grupo de SQA. Segundo, por meio de discussões com auditores e gestores, procurando identificar a sua visão com relação ao processo de auditorias. Como conclusão, idealmente uma solução de verificação de conformidade de processos deveria atender aos seguintes requisitos:

- Ser não intrusiva e objetiva: considera-se que esses dois requisitos sejam diretamente relacionados. Por não intrusivo, se entende um procedimento que não afeta as atividades normais dos colaboradores (um desenvolvedor interrompe sua tarefas para responder a um questionário, por exemplo). Além disso, a pessoa entrevistada pode acabar omitindo ou distorcendo informações por vários motivos. Se o entrevistado não compreender claramente o objetivo daquela entrevista ou questão específica, a qualidade das informações fornecidas pode ser prejudicada. Até mesmo o receio de ter seu desempenho posto em dúvida dependendo da resposta, pode ser um motivo para o entrevistado omitir a visão real que possui sobre o dia-a-dia do projeto em que está envolvido. Tal situação acaba acarretando em subjetividade dos resultados, o que é fator negativo.

- Apresentar indicadores: um fator indispensável em qualquer verificação é a produção de métricas que possam fornecer uma visão quantitativa acerca do objeto avaliado.

- Reduzir e qualificar o trabalho de auditoria: a experiência prática mostrou que a realização de auditorias de SQA em uma organização com diversos projetos de grande porte exige muito trabalho. Em projetos de manutenção que trabalham por demandas, por exemplo, o auditor normalmente escolhe apenas algumas demandas do período para analisar, já que muitas vezes não há tempo hábil para analisar todas. O aumento da eficiência nesse processo é tido também com um aspecto desejável.

- Não interferir significativamente no processo atual: é também um requisito na concepção da solução, que não sejam exigidas mudanças significativas de comportamento e gestão dentro da empresa. Pequenas mudanças normalmente são necessárias para que se obtenha melhores resultados. No entanto, é desejável que tais modificações não afetem significativamente o ambiente atual da organização.

# 4. Estudo sobre conformidade de processos em PDS

Com base no conjunto de requisitos levantados na seção 3.5.1, este trabalho explora como métodos de extração automática de conhecimento podem auxiliar na verificação de conformidade em PDS. Para isto, foi realizado um estudo amplo, abordando três aspectos principais: (i) formato típico de processos definidos em empresas de software, (ii) fontes de dados e formas nas quais registros de execução podem ser utilizados na monitoração de processos e (iii) soluções de verificação de conformidade existentes e tipo de resultados fornecidos a partir de sua aplicação no ambiente de engenharia de software. Embora sendo levantadas algumas questões relativas às visões de recursos e artefatos em processos de software, o presente trabalho dá maior enfoque à análise sob a perspectiva de fluxo de controle (sequência de atividades). Considerando essa visão, primeiramente o OSSP da empresa parceira e o Processo Unificado (RUP 2.0) (33) foram considerados no estudo sobre os processos definidos. O Processo Unificado foi considerado aqui como uma referência com relação ao nível de abstração e formato tipicamente encontrado em modelos de processo em PDS. Além disso, foi realizado um estudo de exploratório com dados de um projeto de manutenção da empresa. Foram executados diversos experimentos com algoritmos de mineração de processos utilizando registros de execução do projeto. Os experimentos permitiram compreender várias questões envolvendo o uso de dados de execução de PDS na aplicação de mineração de processos. Algoritmos de análise de conformidade também foram estudados visando compreender seus requisitos, aplicações e potenciais resultados no auxílio à verificação de conformidade em PDS. As próximas seções reportam o trabalho realizado.

## 4.1 Processos definidos

Para garantir maior controle, previsibilidade e qualidade nos projetos, empresas de software costumam adotar modelos de desenvolvimento. Esses modelos estabelecem métodos sistemáticos que servem como guia para todo o ciclo de vida de um projeto de software. Ao longo dos anos diversos modelos tem sido propostos. Exemplos são: Cascata, Espiral, Evolucionário e Processo Unificado (41). Embora os modelos possuam algumas diferenças, todos se baseiam em uma definição comum. Um processo de software é um conjunto ordenado de tarefas, normalmente distribuídas entre diversas pessoas envolvendo projeto, desenvolvimento ou manutenção de um sistema de software. De forma geral um modelo de desenvolvimento é composto por: fases ou elementos de processo, atividades, produtos e responsabilidades. No momento em que uma empresa adota um modelo, ele deve ser documentado de alguma forma. Tipicamente, tal documentação inclui descrições textuais,

responsabilidades, artefatos envolvidos e diagramas descrevendo os passos de cada fase do processo. Muitas empresas costumam utilizar os chamados *Eletronic Process Guides* (EPG) (36). EPGs proporcionam uma forma eficiente de representar o processo de software. Pode-se navegar facilmente por todos os elementos, visualizando cada fluxo de atividades em notação UML ou alguma outra forma de representação de processos. De uma forma ou de outra, empresas comprometidas com a qualidade e produtividade possuem seus processos formalizados de acordo com o modelo de desenvolvimento adotado.

Soluções de análise de conformidade baseadas em mineração de processos, em sua maioria, assumem a existência de um modelo predefinido, expresso em alguma notação formal(3)(16)(42). Rede de Petri é a forma mais utilizada nesse contexto (1). Redes de Petri são apropriadas à aplicação de técnicas de análise formal pois sua notação provê, além da modelagem visual, uma lógica de execução do sistema modelado. Esse é inclusive um dos principais motivos de sua ampla utilização. Além disso, possui padrões bem definidos, generalizáveis, amplamente conhecidos e de fácil entendimento. Então, para viabilizar a comparação entre um modelo de desenvolvimento de software e sua real execução por meio de mineração de processos deve ser possível obter um workflow em rede de Petri. Ou pelo menos em alguma notação com padrões equivalentes, que possam ser diretamente mapeados. Russel em (44) identifica mais de quarenta padrões de fluxo de workflow, incluindo notações como UML e BPMN. Logo, a obtenção de um modelo de workflow para comparação não deve ser um fator de dificuldade nesse contexto.

Como parte do estudo foram analisados, o conjunto de processos da empresa parceira e o metamodelo de desenvolvimento de software *Rational Unified Process* (RUP) (33). O objetivo foi identificar e comparar o formalismo de workflow adotado e o nível de abstração encontrado nesses processos, tendo em vista a sua aplicação na comparação com dados de execução de um PDS. Primeiramente foram explorados os processos documentados da empresa parceira. Se observou que todo o elemento do OSSP e seus sub-processos possuem diagramas estabelecendo o fluxo de atividades envolvido. Com uma semântica semelhante a diagramas de atividade UML, os diagramas documentados contêm a ordem das atividades, papéis e artefatos envolvidos. Portanto seria possível utilizar tais modelos de workflow como padrão para análise de conformidade.

Um fator a ser considerado, contudo, está relacionado ao nível de abstração dos diagramas. Na sua maioria esses workflows contém atividades em alto nível, sem nenhuma estrutura mais complexa como decisões, iterações e paralelismo. Poucos deles possuem estruturas de decisão modeladas. Entende-se que a comparação desse tipo de estrutura com registros de execução não estaria explorando o potencial que a aplicação de técnicas formais de análise de processos pode ter. O aspecto interessante da descoberta automática de conhecimento está justamente em evidenciar situações que dificilmente podem ser identificadas por meio de análise manual dos dados. Não é considerado relevante por exemplo,

ter como resultado da mineração de processos a confirmação sobre a sequência entre as tarefas de ”desenvolvimento” e ”testes”, o que seria uma informação óbvia. Além disso, os modelos não preveem concorrência entre atividades. Entretanto, em um estudo exploratório com dados de um projeto da empresa (descrito na seção seguinte) se observou que o paralelismo entre atividades é bastante frequente. Logo, é importante que os processos sejam modelados levando em conta no mínimo as três estruturas básicas citadas (paralelismo, decisões e iterações), possibilitando assim obter equivalência entre modelo e registros de execução para viabilizar a comparação provendo resultados mais detalhados e precisos.

Por se tratar de um modelo de desenvolvimento amplamente adotado na indústria atualmente, o formato de workflows do processo unificado também foi analisado. Já nesse modelo, observa-se que as atividades são descritas em menor granularidade, inclusive prevendo paralelismo e decisões. A figura 4.1 mostra o fluxo de implementação do Processo Unificado (33). Evidentemente tal modelagem fornece informações mais precisas acerca do fluxo de atividades esperado no processo. Porém, quanto maior a complexidade do modelo mais difícil se torna garantir a aderência a esse processo sem o auxílio de métodos automatizados. É nesse ponto que técnicas de descoberta de conhecimento podem ser mais efetivas, facilitando a monitoração e controle de processos dinâmicos e complexos como PDS.

## 4.2 Estudo exploratório sobre descoberta de processos

Buscando compreender o tipo de informação tipicamente encontrado em registros de execução de uma empresa de software, foi conduzido um estudo exploratório em um dos projetos da empresa parceira. O ambiente computacional foi explorado a fim de identificar fontes de dados potenciais para a descoberta de conhecimento de processos. Como descrito na seção 3.3, a empresa possui todas as métricas coletadas de sua plataforma computacional integradas em um repositório central. Entre outras informações, a BO armazena atividades desenvolvidas por todos os seus colaboradores. Analisando esse conjunto de dados se observou que, com algumas modificações, os pré-requisitos de um log para mineração de processos (ver seção 2.3.4) poderiam ser satisfeitos. Foi realizado um estudo nesse sentido, buscando avaliar a viabilidade em descobrir relações entre atividades a partir dos registros de esforço da BO. Então, conforme os resultados obtidos, explorar sua aplicação na comparação entre modelo e execução de processos.

Soluções de mineração de processos permitem comparar dois modelos ou ainda um modelo diretamente com um log. São as abordagens de *delta analysis* e *conformance testing*. Mesmo sabendo que é possível comparar um modelo formal diretamente com o log (conformance testing), nesse estudo inicial se decidiu investir na descoberta de processos. A razão foi buscar compreender a estrutura desses dados através da visualização de um

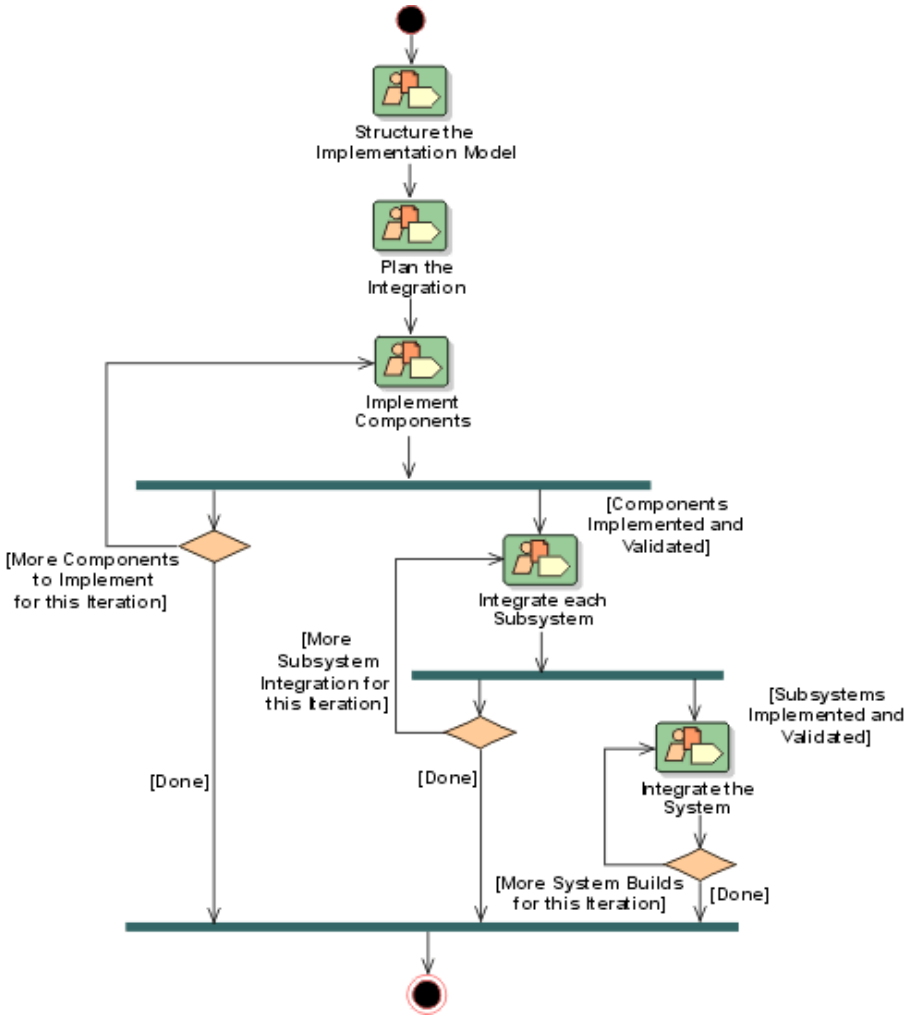


Figura 4.1: Workflow de construção do processo Unificado. Fonte: (33)

modelo de processo. Não seria razoável submeter esses registros a um método de verificação de conformidade sem ter noção da sua semelhança com o modelo de processo predefinido. Enfim, o objetivo foi obter uma visão inicial sobre o tipo de informação obtido, as dificuldades e o trabalho de pré-processamento necessário para esse conjunto de dados, quando submetido a algoritmos de descoberta de processos. Para tanto, foram selecionados dados de um projeto de manutenção (chamaremos de projeto P1) em andamento na empresa. Essa escolha se deu principalmente por quatro motivos:

- Projeto de grande porte, chegando a contar com uma equipe de mais de 50 pessoas. Naturalmente o número de interações entre os participantes e a quantidade de dados produzida são grandes. Além disso, a experiência no grupo de qualidade mostrou ser extremamente difícil obter a visibilidade adequada sobre todos os acontecimentos dentro de um projeto desse porte. Foi inclusive um fator motivador do trabalho.

- Como o projeto foi iniciado em 2005 e continua em andamento, um grande volume de dados foi produzido nesse período. Apenas a base de registros de atividades anal-

isada continha mais de cem mil linhas na época da realização do estudo, registradas durante quatro anos de projeto. Sabe-se que a abundância de dados é um fator chave na aplicação de qualquer técnica de descoberta de conhecimento (28).

- Projetos de manutenção e melhoria são bastante comuns na empresa. No período da pesquisa, metade dos projetos em andamento eram desse tipo.
- O fluxo iterativo é uma característica comum em projetos de manutenção. Conjuntos de demandas solicitadas pelo cliente regularmente, são normalmente agrupadas em ordens de serviço (OS). Por sua vez, um conjunto de OS forma uma versão de software a ser entregue ao cliente. Cada OS passa por uma etapa de estimativas e, em seguida, passa pelas fases de projeto e desenvolvimento (divida entre equipes cliente e servidor). Após a etapa de testes, um grupo de OS é agrupado em um *build* e disponibilizado como uma versão do produto. Tal estrutura favorece a representação dos registros de execução sob a forma <*instância,atividade*> considerada nos algoritmos de mineração de processos. Tanto o identificador de OS quanto a versão podem identificar uma instância (iteração) do processo de manutenção.

### 4.2.1 A base de registros de atividades

Dentro da BO, grande parte da tarefas desenvolvidas nos projetos são armazenadas em uma base de dados específica. A figura 4.2 demonstra a estrutura dessa base. Nela, cada colaborador registra suas atividades no projeto, dentro de uma classificação pré-definida (tabela TIPO_ATIVIDADE). Essa tabela contém 57 tipos de atividades, abrangendo todo o ambiente de execução do projeto. Exemplos de atividades são: planejamento, teste unitário, liberação, backup e reuniões. Os registros são feitos efetivamente na tabela ATIVIDADE. Cada linha dessa tabela possui, entre outras informações, o grupo envolvido, o responsável, o tipo de atividade base, projeto e a duração da atividade. O atributo Grupo relaciona a equipe envolvida na atividade. Adicionalmente, uma atividade do projeto está relacionada a um Tipo de Atividade Base (Trabalho, Retrabalho, Revisão e Qualidade). Atividades não são necessariamente atreladas a uma ordem de serviço, podendo pertencer ao escopo de uma versão.

Foi realizada uma série de experimentos sobre esse conjunto de dados. Um relato da primeira experiência realizada foi publicado em (18). Uma versão estendida desse trabalho foi submetida ao *International Journal of Business Process Integration and Management* e aguarda avaliação. Para a execução dos experimentos foi utilizada a plataforma Process Miner (26). A ferramenta possui suporte completo a mineração de processos, abrangendo todas as perspectivas de mineração (fluxo de controle, organizacional e informação). O ProM é uma ferramenta acadêmica, livre, mantida por um dos maiores grupos de pesquisa

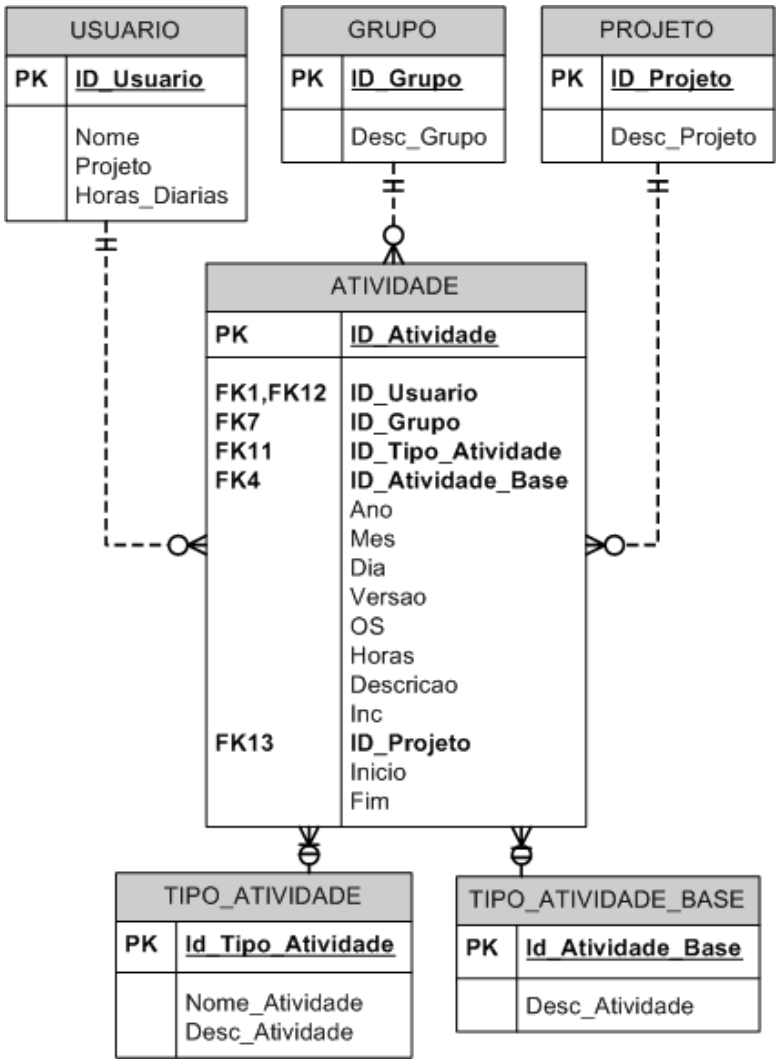


Figura 4.2: Diagrama de dados da base ponto atividade

na área (26), que integra grande parte dos algoritmos de mineração de processos propostos no meio acadêmico, tanto para descoberta de processos, quanto para análise de conformidade (48) (26). O ProM é implementado em uma estrutura de *plugins*, facilitando a inclusão de novas soluções. Outra facilidade é a interoperabilidade. É possível importar dados de diversas fontes. É utilizado um formato de entrada padrão MXML (47) para representação de processos, provendo então um *plugin* de conversão de diversos tipos de logs para esse formato. O formato XML voltado à representação de processos, visa ser genérico o suficiente para suportar logs provenientes da maioria dos sistemas de workflow encontrados atualmente no mercado. Ainda é possível importar dados de um banco de dados MS Access, o que foi bastante útil para a realização dos experimentos. A base de métricas armazenada em ambiente SQL Server foi exportada para uma base MS Access, facilitando a conversão dos dados para MXML. A figura 4.3 mostra o diagrama de relacionamento das tabelas utilizadas no formato MXML a partir de um banco de dados.

A tabela *Audit_Trail_Entries* contém os registros de execução efetivamente. Todas as instâncias de processo devem ser listadas sem repetição na tabela *Process_Instances*. Além dessas duas tabelas obrigatórias, dados adicionais sobre o processo podem ser inseridos nas tabelas *Data_Attributes_Audit_Entries* e *Data_Attributes_Process_Instances*. Os dados armazenados nessas duas últimas tabelas servem principalmente para a análise sob a perspectivida de *informação*, descrita na seção 2.3.2.

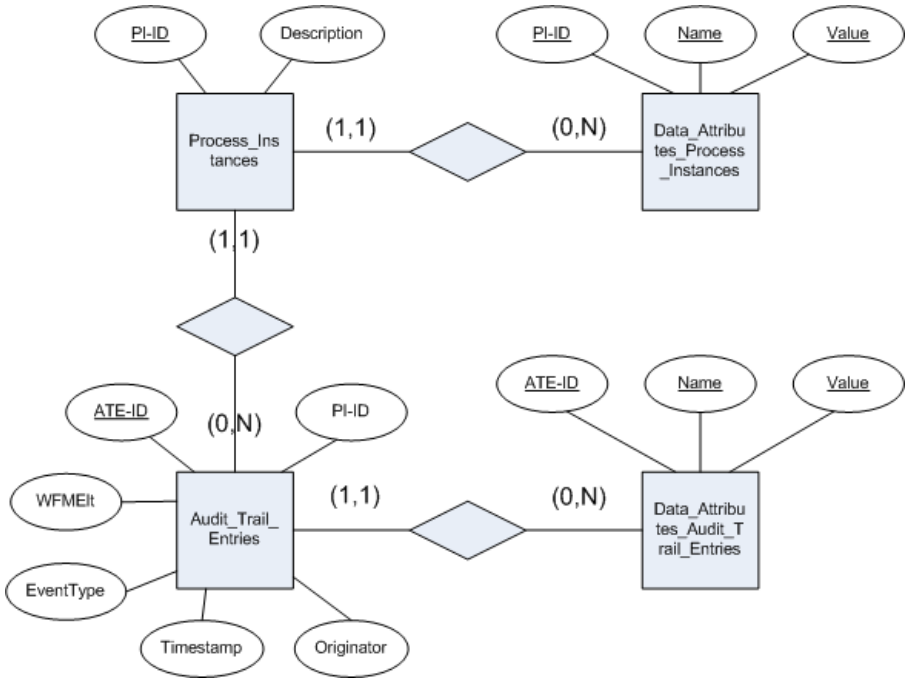


Figura 4.3: Diagrama E-R do formato MXML

### 4.2.2 Estudo dos algoritmos

Para realizar os testes se fazia necessário selecionar algoritmos de descoberta de modelos apropriados para as propriedades dos dados utilizados. Então se buscou na literatura as abordagens existentes procurando compreender suas características. De forma geral, cada algoritmo busca resolver algum problema específico envolvendo a mineração de processos (4) (12). Entre esses problemas se pode citar o tratamento de ruído, dados pouco estrurados e incompletude no log. Durante o trabalho de preparação, foram observadas algumas particularidades nos dados que serviram de parâmetro para a escolha. Um aspecto importante identificado foi a presença de dados incorretos (ruído). Alguns campos importantes para a realização da mineração não possuiam restrição no preenchimento. Consequentemente, se encontrou uma quantidade considerável de registros com campos não preenchidos ou preenchidos incorretamente. Lidar com informações incorretas é um fator considerado fundamental para a escolha do algoritmo nesse ambiente. Outro aspecto levado em conta é o tipo do processo. A bibliografia da área divide processos em estrutura-

dos e não estruturados (38). Existem algoritmos voltados a cada um desses formatos. Em processos não estruturados, o número de caminhos possíveis geralmente é muito grande. Isso torna difícil a obtenção de modelos simples e que ao mesmo tempo representem com fidelidade o processo. Abordagens voltadas a processos estruturados costumam ser aplicadas a log de sistemas de workflow, tratando-se de processos mais bem comportados. Por não se tratar de registros produzidos por um sistema governado por um modelo de workflow, a base de atividades se enquadra como um processo não estruturado.

Outro aspecto que caracteriza algoritmos de mineração de processos diz respeito à completude dos dados. Na prática, um log tipicamente não contém todas as sequências de eventos possíveis no processo. Modelos tem de ser inferidos com base em um subconjunto de sequências determinado pelo número de instâncias presente no log. Para o caso dos experimentos realizados, outra característica desejável é com relação à flexibilidade dos resultados produzidos. Certos algoritmos permitem interação do usuário por meio do ajuste de parâmetros, obtendo assim diferentes visões sobre o mesmo log. A idéia era explorar diferentes visões do modelo a fim de identificar as relações mais interessantes entre as atividades do processo. A tabela 5.1 mostra um comparativo entre quatro algoritmos estudados, baseado nas dimensões de interesse para o trabalho. Tais algoritmos foram selecionados por serem os mais conhecidos na literatura.

O algoritmo $\alpha$ é considerado o pioneiro na descoberta de processos. Sua abordagem básica para identificação de relações entre eventos é utilizada pela maioria dos algoritmos atuais. O algoritmo $\alpha$++ também foi proposto, procurando eliminar algumas limitações da solução anterior, sendo capaz de detectar dependências implícitas e laços curtos no log. Entretanto, ambas as soluções possuem a mesma restrição. Elas consideram um log sem erros e completos (cada relação tem de aparecer pelo menos uma vez para ser detectada) para a descoberta do modelo. Porém na prática tal situação não acontece. Logs são geralmente incompletos e contém erros. Dessa forma, atualmente algoritmos tem adicionado tratamentos estatísticos para lidar com esse problema. As soluções Heuristics Miner (50) e Fuzzy Miner (27) são dois exemplos de aplicação de métodos estatísticos. As suas abordagens são bastante semelhantes, sendo capazes de lidar eficientemente com processos pouco estruturados, incompletos e contendo ruído. Eventos e relações entre esses eventos são construídos com base na frequência em que aparecem no log. O objetivo das duas soluções é induzir o comportamento mais significativo do processo, mesmo a partir de registros incompletos e incorretos. Essas duas soluções se mostram interessantes para o tipo de dados explorado no contexto de PDS, justamente por envolver dados pouco estruturados, incompletos e imperfeitos.

Como o nome sugere, o algoritmo Heuristics Miner (50) se utiliza de certas heurísticas para obter o fluxo mais relevante do processo aplicando métodos estatísticos. O ponto de partida do algoritmo é a construção do chamado grafo de dependências. Esse grafo é

Tabela 4.1: Comparação entre algoritmos de mineração de processos

| Algoritmos | **Ruído e incompletude** | **Tipo de processo** | **Refinamento do modelo** | **Disponibilidade** |
|---|---|---|---|---|
| $\alpha$ (8) | assume log sem ruído e completo | estruturado | não possui | implementado no ProM |
| $\alpha$ ++ (51) | assume log sem ruído e completo | estruturado | não possui | implementado no ProM |
| Heuristics Miner (50) | lida com ruído e incompletude (abordagem estatística) | não estruturado | Usuário provê heurísticas que modificam o modelo obtido | implementado no ProM |
| Fuzzy Miner (27) | lida com ruído e incompletude (abordagem estatística) | não estruturado | Permite alterar o modelo por meio de parâmetros configuráveis em tempo real | implementado no ProM |

obtido a partir de uma tabela que relaciona todas as atividades, calculando a frequência relativa de precedência entre cada uma. É utilizado um valor limiar para filtrar apenas as correlações mais relevantes. Primeiramente são obtidas as frequências relativas de precedência entre todos os eventos. O maior dentre esses valores é tomado como uma heurística de melhor caso. Então, é possível definir um limiar denominado *relative to best threshold*, para estabelecer o nível de filtragem que o algoritmo fará com base naquele primeiro valor de frequência relativa. É aplicado ainda um tratamento para concorrência. É construída uma tabela contendo todas as atividades com suas relações de concorrência (AND/OR-split/join). A tabela 4.2 mostra o formato dessa tabela para um possível log com cinco atividades diferentes. A entrada de atividades iniciais e saída de atividades finais são expressadas como zero. A partir de uma série de operações lógicas utilizando essa tabela e as relações do grafo de dependências, o algoritmo extrai todas as relações de paralelismo entre as atividades. Alguns indicadores apresentados no próprio modelo também fornecem pistas sobre o comportamento do processo. São mostradas a frequência de cada atividade, de cada relação e também o índice de correlação entre cada par de atividades. A semântica de concorrência também é mostrada no modelo final, como na figura 4.5.

A abordagem do algoritmo Fuzzy Miner é bastante semelhante à solução anterior. Am-

Tabela 4.2: Ilustração da matriz de concorrência do algoritmo Heuristics Miner. Fonte: (50)

| ACTIVITY | INPUT | OUTPUT |
|---|---|---|
| A | $\oslash$ | (B$\vee$ E) $\wedge$ (C$\vee$ E) |
| B | A | D |
| C | A | D |
| D | (B$\vee$ E) $\wedge$ (C$\vee$ E) | $\oslash$ |
| E | A | D |

bos utilizam frequências para gerar eventos e correlações entre eles. Entretanto, o Fuzzy Miner se diferencia em dois aspectos. Primeiro, é empregada uma abordagem multi-perspectiva para correlacionar eventos (analisar semelhança dos atributos dos eventos). Segundo, são empregadas ferramentas de customização dos resultados para simplificação de modelos de processos pouco estruturados. Segundo os autores, a inteligibilidade dos resultados é um grande problema na descoberta desse tipo de processos. Como existem muitas conexões, os modelos tornam-se extremamente confusos para o entendimento humano. O problema é conhecido como geração de modelos *spaghetti-like* (5). Então, são aplicados conceitos retirados da cartografia para simplificação dos modelos. Tais conceitos são: agregação, abstração, ênfase e customização. Informações em maior nível de detalhe aparecem agrupadas no resultado (ex.: mapa-mundi mostra cidades apenas como pequenos pontos). Informações irrelevantes para o contexto são abstraídas, ao contrário do comportamento mais relevante que é enfatizado no modelo (ex.: mapa rodoviário enfatiza as auto-estradas e não os pontos turísticos de uma região). Assim como em mapas, os autores consideram que não existe um único modelo universal. Diferentes visões são possíveis, dependendo do contexto. O conceito de customização está relacionado à atribuição de valores ajustáveis, proporcionando diferentes visões do modelo de acordo com o propósito de análise. O algoritmo produz o modelo do processo baseado em alguns critérios. Tais critérios de decisão são suportados por duas métricas básicas: significância e correlação. A significância pode ser determinada, tanto para eventos quanto para relações de precedência binária entre esses eventos. Ela mede a importância relativa de um determinado comportamento. A significância é baseada na frequência, ou seja, quanto mais frequente é um evento ou uma correlação entre dois eventos maior será sua significância. Por outro lado, a correlação é calculada apenas para relações de precedência entre eventos. Ela expressa o quão proximamente relacionados são dois eventos. O Fuzzy Miner possui várias métricas para o cálculo da correlação, todas baseadas na semelhança de atributos contidos no log. Uma das métricas mede o grau de semelhança entre identificadores de dois eventos distintos. Por exemplo, as atividades *analisar_pedido_cliente* e *aprovar_pedido_cliente* são consideradas fortemente correlacionadas devido à semelhança nos nomes de seus identificadores. A partir dessas duas métricas o modelo de processo é construído de forma que:

- Comportamento altamente significativo é preservado.
- Comportamento menos significativo, mas altamente relacionado é agregado.
- Atividades pouco frequentes e pouco correlacionadas são omitidas (abstração).

### 4.2.3 Análise e pré-processamento dos dados

A etapa de preparação é parte importante em qualquer processo de descoberta de conhecimento sobre conjuntos de dados. Nesta fase, é feita uma análise criteriosa a fim de eliminar informações desnecessárias e também estruturar os dados de forma a serem melhor interpretados pela ferramenta de mineração (28). No caso do estudo realizado, a preparação consistiu em duas etapas: (i) seleção e pré-processamento dos dados (ii) conversão do banco de dados para o formato MXML. Os experimentos foram realizados sob três perspectivas: atividades ligadas a OS, versão e a um único processo. Cada um deles exigiu um procedimento diferente de análise e pré-processamento dos dados, o que foi de grande valia para o entendimento sobre a estrutura e o comportamento dos dados de métricas, quando analisados sob a visão de mineração de processos.

#### Análise de OS e versões

Como primeira exploração do registros, experimentos foram realizados relacionando as atividades com a iteração relacionada (OS e versão). Primeiramente, foram selecionadas as atividades consideradas mais relevantes no processo e com comportamento mais previsível. Foram descartadas atividades do tipo folga, estudo e viagem por não possuirem um momento específico para ocorrer. Embora essas atividades possam ser consideradas no planejamento, elas não fazem parte dos processos organizacionais formalizados. Por isso, foram descartadas. A seleção foi feita com base na descrição de cada atividade além do auxílio de colaboradores com conhecimento sobre o contexto do projeto. A representação de um conjunto de eventos para mineração de processos supõe pelo menos três informações: identificador da instância do processo, nome da atividade e *timestamp*. Alguns algoritmos também utilizam o responsável (*originator*) e possivelmente um campo de estado de execução da atividade. Por isso, esses últimos campos foram também adicionados ao log. Os dados foram selecionados da seguinte forma (ver figura 4.2):

- Instância: tanto o campo *OS* quanto a *Versão* poderiam servir como identificador para instância. No entanto, identificou-se que existem atividades específicas pertencentes a cada um desses dois contextos. Sendo assim, foram gerados dois logs distintos sendo, um com OS como instância e outro com o campo Versão (cada um com suas respectivas atividades).

- Atividade: foi utilizada a descrição da atividade diretamente.
- Timestamp: Como pode ser observado na figura 4.2, o instante de execução das atividades é dividido em ano, mês, dia. Além disso, não existe um horário de início e fim, somente a duração. O timestamp no formato MXML (*dd–mm–aa hh:mm:ss*) foi gerado concatenando os três primeiros campos e adicionando uma hora padrão de início para todas as entradas.
- EventType: foi adicionada uma string artificialmente (*completa*).
- Originator: coluna *ID_Usuario* da tabela USUARIO.

Tendo selecionado os atributos, a etapa de pré-processamento consistiu na implementação de consultas SQL para replicar as colunas necessárias na tabela principal (Audit_Trail_Entries) da estrutura MXML. Adicionalmente foi realizado uma etapa de limpeza dos dados, buscando eliminar registros incompletos ou com preenchimento incorreto. Sabe-se que quanto menor o volume de ruído nos registros, melhores são os resultados produzidos pelos algoritmos de mineração.

Resultados

A figura 4.4 mostra o modelo produzido pelo algoritmo Fuzzy Miner. Cada nodo contém o seu valor de significância. Os arcos possuem respectivamente os valores de significância e correlação. O modelo realmente sugere alguns comportamentos coerentes, segundo informações retiradas dos processos organizacionais e confirmadas pelos colaboradores do projeto. A atividade *OS* é inicial. De acordo com a descrição dessa atividade, ela envolve o trabalho nas solicitações do cliente (análise de documentos técnicos, e-mails, etc.). É esperado que ela seja realizada normalmente no início da demanda. O seu valor de significância igual a 1 também mostra que é uma atividade muito frequente e repetitiva, o que é confirmado pelos altos valores de significância e correlação do conector iterativo (loop). Outro fato a ser observado é a conexão de praticamente todas as atividades com a OS. O motivo é a inserção de várias entradas dessa atividade no decorrer do projeto, por ser uma realizada sob demanda. Como esse comportamento é frequente, o algoritmo o identifica.

Em seguida observa-se as atividades *Estimativa*, *Setup-Dev* e *Teste Unitário* conectadas. Esse comportamento também é consistente com o processo. Segundo informações dos colaboradores do projeto, o trabalho de estimativa é realizado logo após a definição da documentação de projeto da demanda. A próxima atividade *Setup-Dev* recebe esse nome porque, antes do início do desenvolvimento, é utilizada uma ferramenta de geração de pseudo-código a partir da documentação de projeto. Nota-se também a atividade *Construção* ligada a atividade anterior, o que é bastante coerente. A execução de testes unitários de código é realizada pelo desenvolvedor após a conclusão de cada unidade de

código. A relação entre essas duas atividades também aparece no modelo. Por fim, há uma correlação entre as atividades *Caso de Teste* e *Caso de Teste - Retrabalho*. Essa sequência sugere a correção de defeitos encontrados durante a execução inicial dos casos de teste. Cabe salientar que este modelo não foi obtido como primeiro resultado da execução. A interface do *plugin* permite ajustar todos os parâmetros do algoritmo, reconstruindo o modelo imediatamente. Então, esses valores foram modificados empiricamente até chegar-se a um resultado considerado consistente com o processo real.

Na figura 4.5 é apresentado o resultado do algoritmo Heuristics Miner. Mesmo com algumas diferenças do modelo anterior, este algoritmo também é capaz de capturar características relevantes. Isso reforça a tese de que os dois algoritmos realmente conseguem capturar o comportamento de processo pouco estruturados. O Heuristics Miner também provê duas métricas, que são mostradas no modelo resultante. A primeira expressa um valor de confiabilidade calculado para cada relação de precedência. O segundo valor apresentado é uma contagem direta do número de vezes em que o evento/conexão apareceu no log.

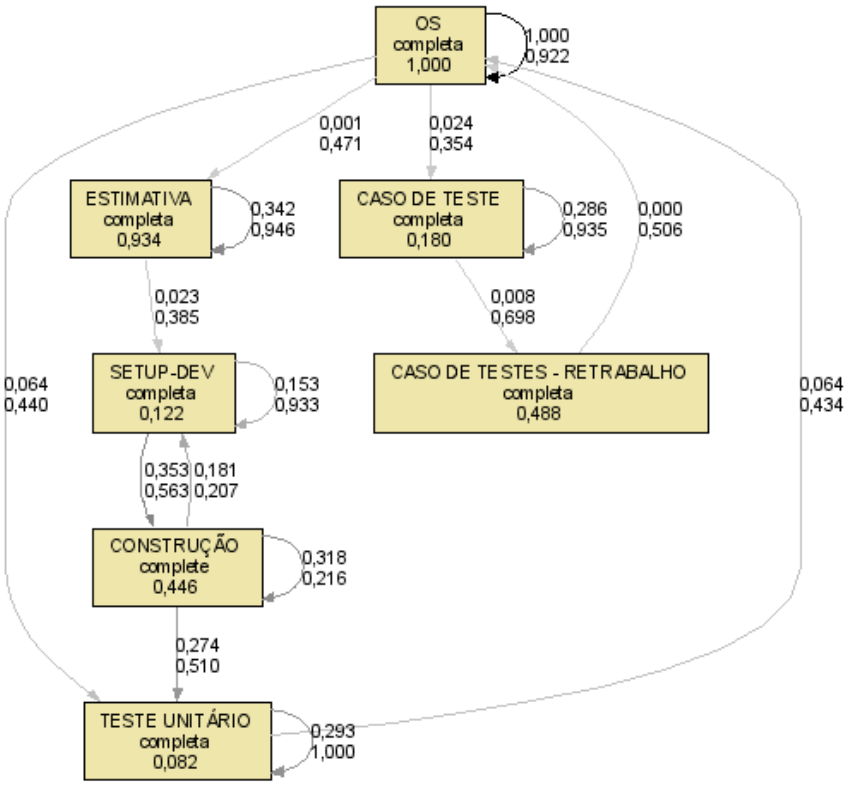


Figura 4.4: Modelo produzido pelo algoritmo Fuzzy Miner - atividades ligadas a OS

Também foi realizado um experimento com atividades relacionadas a uma versão do produto (figuras 4.6 e 4.7). Nesse caso, além de filtrar as atividades, a única modificação foi o adotar o campo *Versão* da tabela ATIVIDADE como instância de processo. Os resultados também foram relativamente consistentes com o processo real. A atividade *SPP* está rela-

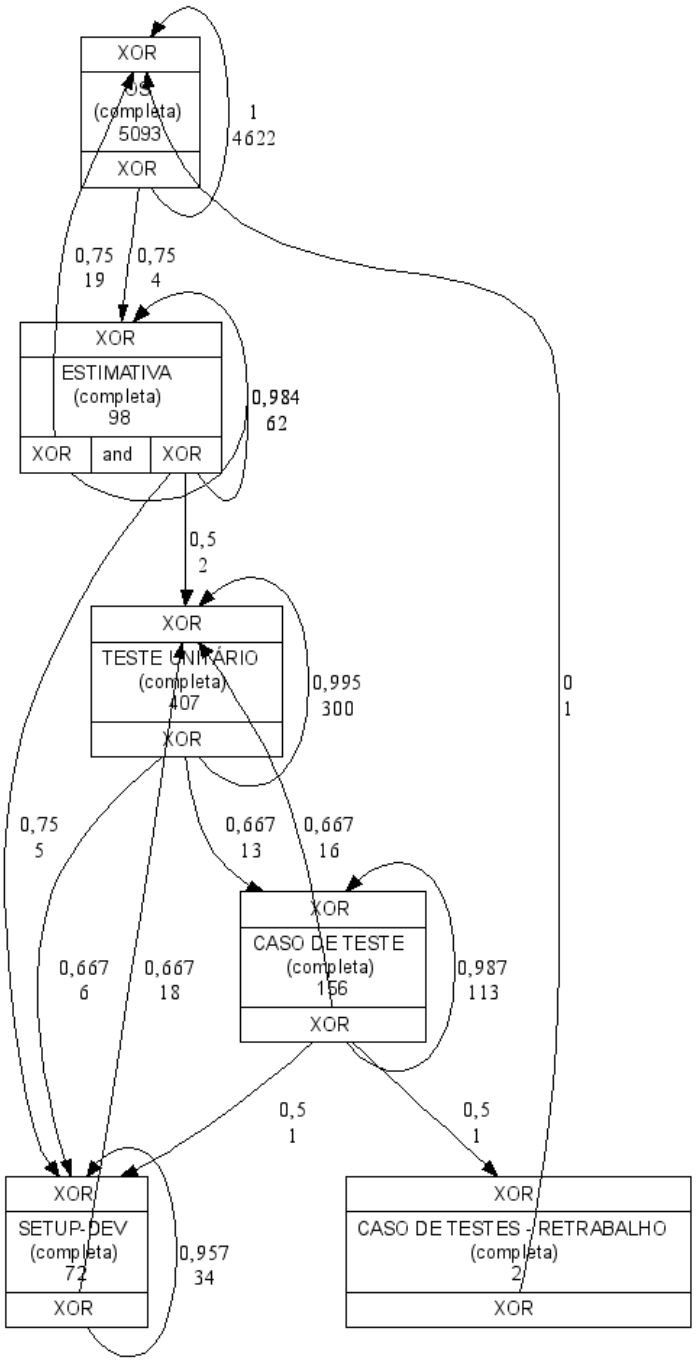


Figura 4.5: Modelo produzido pelo algoritmo Heuristics Miner - atividades ligadas a OS

cionada ao planejamento inicial da versão. Logo após esse planejamento é iniciada a construção do plano de testes. No entanto, nos dois algoritmos a atividade *Peer Review Plano de Teste* apareceu antes de *Plano de Teste*. O motivos do aparecimento de atividades com ordem invertida e conexões aparentemente sem sentido são discutidos posteriormente, nas conclusões do estudo. Logo, vale salientar que as métricas apresentadas nos modelos não podem ser considerados confiáveis, dada a natureza experimental/empírica dos experimentos. Embora tenham sido identificados comportamentos interessantes, muitas conexões inconsistentes apareceram. Contudo, os resultados serviram ao propósito inicial. O objetivo foi justamente realizar uma análise exploratória acerca das características do conjunto de dados e possibilidades de extração de conhecimento considerando uma perspectiva de fluxo de controle de processos. Um estudo em maior profundidade seria necessário para obter formalmente os valores adequados de parâmetros dos algoritmos, podendo assim compreender com maior precisão as métricas produzidas.

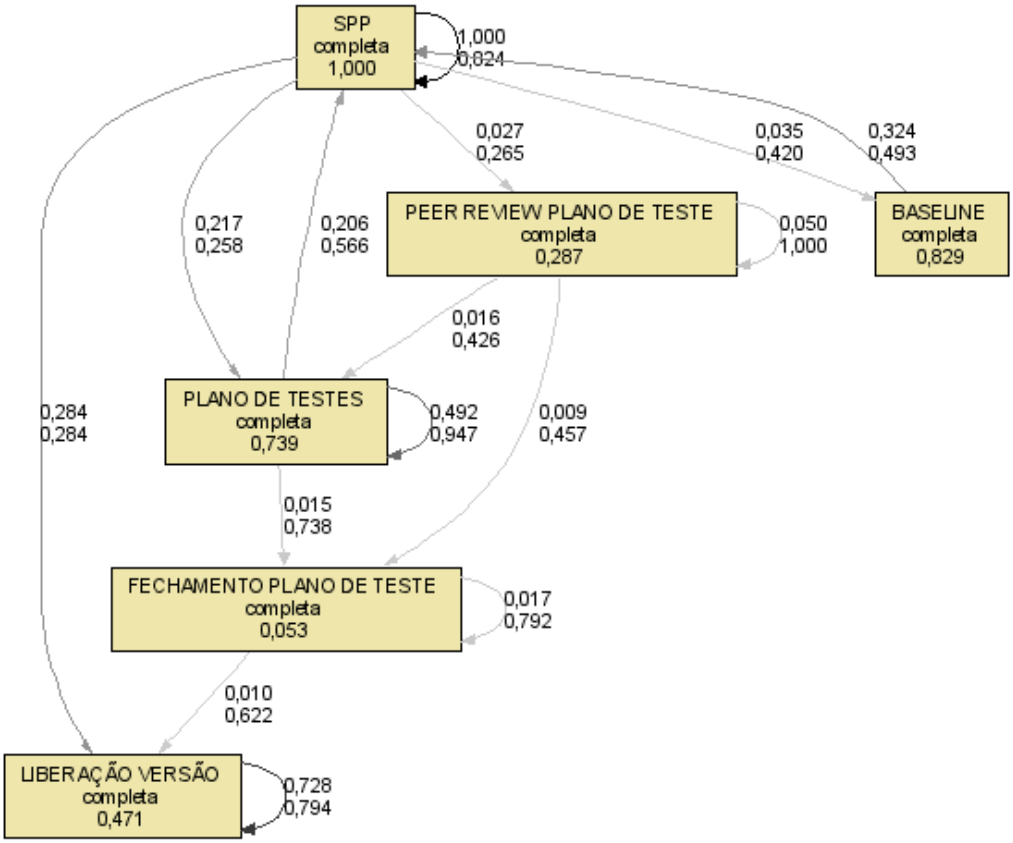


Figura 4.6: Modelo produzido pelo algoritmo Fuzzy Miner - atividades ligadas a versão

Análise de uma fase de projeto

Os experimentos anteriores foram realizados como uma primeira aproximação, buscando identificar padrões relevantes no processo. O escopo de instâncias definido para

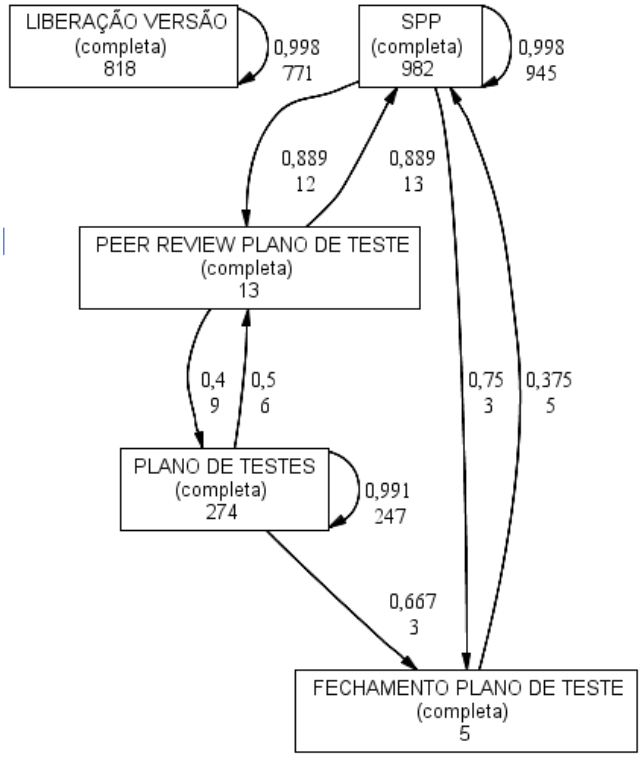


Figura 4.7: Modelo produzido pelo algoritmo Heuristics Miner - atividades ligadas a versão

a mineração foi relacionado simplesmente às duas unidades principais de iteração encontradas nos projetos de manutenção da empresa parceira: versões e OS. Apenas as atividades consideradas de menor relevância para a análise do processo foram retiradas. No entanto se percebeu que uma abordagem mais objetiva seria necessária, visando obter resultados mais próximos dos processos predefinidos para viabilizar a comparação entre ambos. Então foi realizado um terceiro experimento concentrando-se somente em uma fase de projeto, sendo escolhida a fase de testes. Essa fase foi escolhida por dois motivos principais. Primeiro, por ser umas das etapas com maior atividade no ciclo de vida do projeto. Segundo, pelo fato de seu workflow não ser tão abstrato como alguns dos outros processos analisados. Além de expressar as atividades em menor granularidade, esse workflow inclui uma estrutura de decisão na parte de retrabalho. Tinha-se como objetivo identificar no processo minerado padrões semelhantes com os definidos no modelo. Nessa exploração se identificou algumas características que se tornam importantes para melhorar a qualidade dos achados.

Primeiro, se observou que nos registros uma tarefa não possui explicitamente um início e fim. A cada vez que um colaborador trabalha na tarefa, ele registra o tempo em que esteve envolvido na sua execução. Dessa forma, a mesma tarefa pode conter um número indefinido de entradas no registro. De fato os algoritmos de mineração identificam tal situação, representada como um laço para a própria atividade repetida. Isso pode ser

observado nos resultados anteriores. Porém, esse tipo de relação não é coerente com a representação natural de eventos em processos. Tarefas normalmente constituem-se de um estado de início e fim. Tal forma de representação permite modelar explicitamente a concorrência em processos, onde atividades possuem um período duração determinado. Mesmo analisando superficialmente os dados de execução do projeto P1, se pode evidenciar a presença frequente de atividades executadas em paralelo. Sendo assim, é importante que esse comportamento possa ser devidamente rastreado. A alternativa mais óbvia para modelar os diferentes estados das atividades seria inserí-los na coluna *EventType* da tabela *Audit_Trail_Entries*. Entretanto, logo se identificou que a maioria dos algoritmos de mineração não utilizam esse campo para a análise de concorrência. Então, os estados de início e fim das atividades tem de ser inseridos juntamente com o identificador da própria atividade. Isso foi feito da seguinte forma: para uma mesma instância, foram produzidos eventos de início e fim para todas as atividades. Para aquelas que possuíam entradas repetidas, o estado de início foi inserido na atividade com data mais antiga e o fim naquela com data mais nova. As entradas intermediárias foram eliminadas. Atividades com uma única entrada foram duplicadas, apenas concatenando o texto ”inicio” e ”fim” em cada uma das cópias. A figura 4.8 mostra um trecho do log produzido.

Feito o procedimento descrito, a próxima etapa envolveu a seleção das atividades relacionadas à respectiva fase. A seleção foi feita analisando a semelhança entre o nome da atividade no registro e no modelo definido. Foram selecionadas inicialmente quatro atividades: *Plano de Testes*, *Caso de Teste*, *Caso de Testes-Retrabalho* e *Liberação Versão*. Contudo, o workflow definido contém dez atividades. Então a estratégia foi procurar informações adicionais que permitissem mapear as atividades faltantes. Conforme informação de colaboradores do projeto, um campo *Descrição* era utilizado para o registro de comentários adicionais sobre a realização das tarefas. Esse campo foi então explorado, sendo identificados certos padrões que foram utilizados para aumentar o detalhamento do processo. Na verdade, na atividade *Caso de Teste* se encontrou informações mais interessantes. Palavras do tipo criação, relatório, alteração, execução eram bastante frequentes. Tais identificadores foram também concatenados à atividade principal, aproximando com isso o nível de abstração dos registros ao do modelo predefinido. Os modelos resultantes da execução dos algoritmos Heuristics Miner e Fuzzy Miner são apresentados nas figuras 4.9 e 4.10.

### Resultados

Os resultados de ambos algoritmos para o processo de testes apresentam padrões bem definidos. No entanto, um fato interessante a se observar é que tais padrões não correspondem exatamente ao fluxo documento no OSSP. Essa pode ser uma evidência que comprova

| ATE-ID | PI-ID | WfMElt | EventType | Timestamp | Originator |
|---|---|---|---|---|---|
| 33868 | 09.00.00 | PEER REVIEW CT-INICIO | completa | 14/06/2006 09:30:00 | 50 |
| 33891 | 09.00.00 | CASO DE TESTE-INICIO | completa | 22/06/2006 09:00:00 | 50 |
| 33734 | 09.00.00 | CRIAÇÃO/ALTERAÇÃO CT-INICIO | completa | 03/07/2006 12:00:00 | 50 |
| 41798 | 09.00.00 | CASO DE TESTE-REPORTAR TESTES-INICIO | completa | 01/08/2006 17:00:00 | 41 |
| 43725 | 09.00.00 | PLANO DE TESTES-INICIO | completa | 03/08/2006 17:00:00 | 14 |
| 44964 | 09.00.00 | CRIAÇÃO/ALTERAÇÃO CT-FIM | completa | 09/08/2006 18:00:00 | 50 |
| 45755 | 09.00.00 | PEER REVIEW CT-FIM | completa | 16/08/2006 09:20:00 | 50 |
| 45943 | 09.00.00 | CASO DE TESTE-FIM | completa | 18/08/2006 14:00:00 | 32 |
| 120517 | 09.00.00 | CASO DE TESTE-REPORTAR TESTES-FIM | completa | 01/08/2006 18:00:00 | 41 |
| 120516 | 09.00.00 | PLANO DE TESTES-FIM | completa | 03/08/2006 18:30:00 | 14 |

Figura 4.8: Trecho do registro preparado para mineração

de fato a existência de diferenças entre o comportamento esperado e a realidade de execução dos processos. Pode-se constatar, por exemplo, que a primeira atividade do processo é normalmente *Criação de Caso de Teste*. Inclusive a significância da atividade é bastante alta nos dois algoritmos, indicando que esse comportamento acontece na maioria dos casos. O que não corresponde de forma precisa ao modelo definido, que estabelece como primeira atividade o planejamento de testes. Analisando o modelo produzido pelo algoritmo Fuzzy Miner, podemos ver que a atividade *Plano de Testes* na verdade ocorre em paralelo com parte do processo. Já alguns outros comportamentos são coerentes com o esperado. Após a execução dos casos de teste (atividade Caso de Teste), aparecem as atividades de retrabalho em construção e integração (*Caso de Teste - Retrabalho - Constr/Integr*). Este também é um comportamento previsto no modelo definido.

### 4.2.4 Conclusões sobre o estudo

A extração de conhecimento sobre processos a partir da base de métricas se mostrou promissora. No entanto, durante a preparação dos dados e após a análise dos modelos, ficou claro que algumas características desses dados devem ser observadas a fim de obter melhores resultados na mineração. Os principais pontos são enumerados a seguir:

- Análise de processos x análise de métricas: relevância dos dados

  Uma questão relevante a ser considerada diz respeito à importância de cada informação para um ou outro contexto (análise de métricas x mineração de processos). Considerando o compartilhamento da mesma fonte de dados, um equilíbrio deve ser atingido para satisfazer os requisitos, tanto da análise de métricas quanto para análise de processos. No estudo observou-se por exemplo, que alguns campos como hora de início e fim não possuiam restrições de preenchimento implementadas. Consequentemente, o conteúdo desses campos apresentava problemas. Campos em branco e registrados de forma incorreta eram bastante comuns. Normalmente, algoritmos de mineração de processos utilizam um *timestamp* para ordenar os eventos dentro de

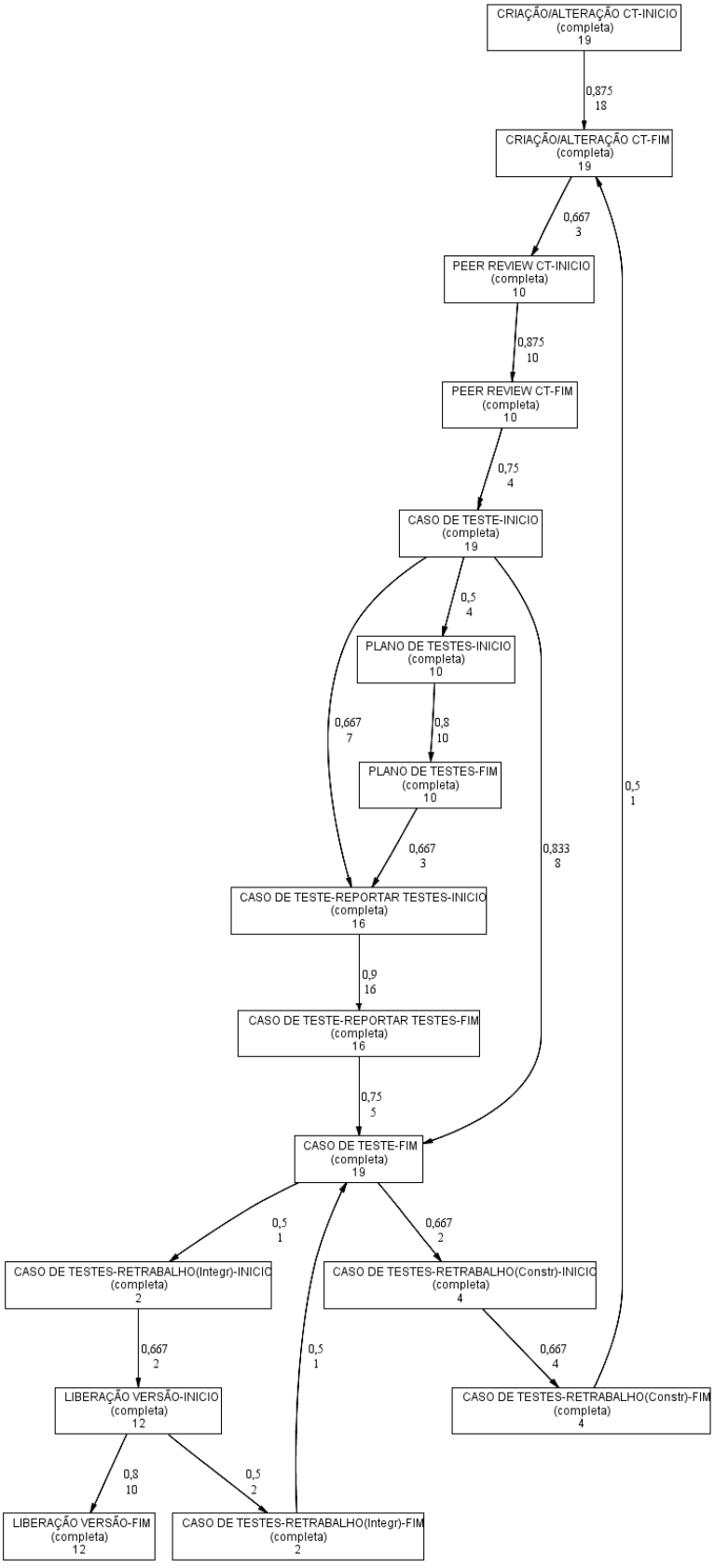


Figura 4.9: Modelo produzido pelo algoritmo Heuristics Miner - processo de testes

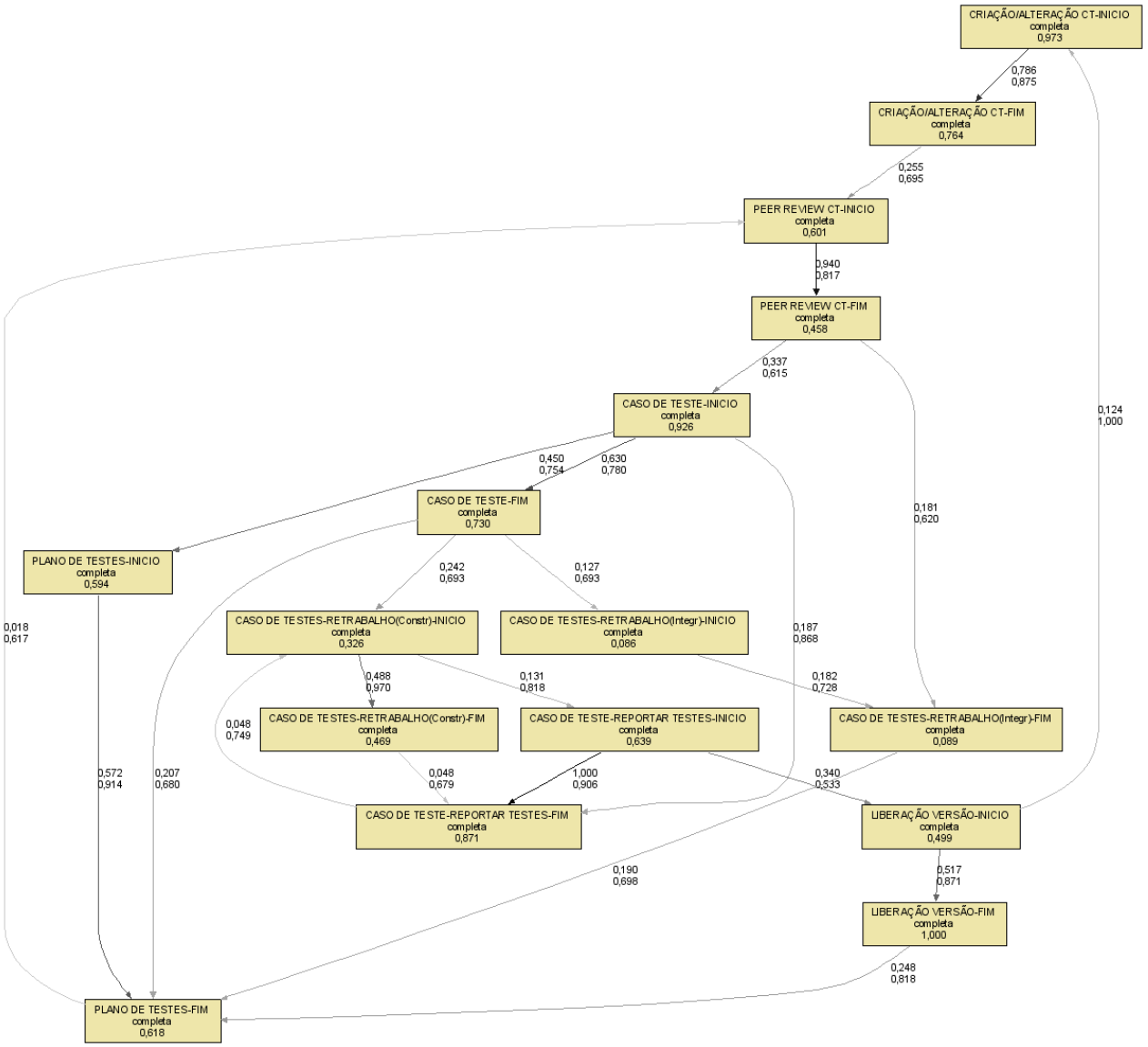


Figura 4.10: Modelo produzido pelo algoritmo Fuzzy Miner - processo de testes

uma instância do log. Entretanto, nem sempre era possível encontrar o momento exato de execução das atividades nos registros da BO. Embora existissem os campos início e fim, se descobriu posteriormente que eles não eram relevantes para a análise das métricas. A informação de maior interesse é a quantidade de horas trabalhadas. Valor este que é armazenado diretamente na coluna *Horas*, como pode ser observado na figura 4.2. O registro completo de *timestamp* é fundamental para a inferência de ordenação entre atividades. Algoritmos de mineração de processos detectam ordenação e paralelismo, pressupondo registros de atividades de duas formas: atômicas (execução instantânea) ou tendo início e fim. Para atividades atômicas, são consideradas paralelas duas atividades que aparecem em qualquer ordem no log. Ainda é possível modelar paralelismo explicitamente dividindo atividades em início e fim. Con-

forme a granularidade assumida no processo, a informação de tempo pode ser mais ou menos completa. No entanto, para PDS onde várias atividades de curta duração são realizadas durante um único dia, um *timestamp* completo deve ser registrado para que se obtenha resultados mais precisos.

- Processo definido x executado: granularidade de atividades

  Outro fator observado está relacionado à granularidade e escopo das atividades. Para possibilitar a comparação entre os processos definido e executado, as atividades no modelo e nos registros devem ser equivalentes. Deve existir um mapeamento preciso entre as duas partes, tanto na identificação quanto no nível de abstração das atividades. No estudo de caso se identificou que grande parte das atividades são registradas em um nível de detalhe diferente daquele encontrado nos processos definidos. Na verdade, esse problema é esperado pelo fato de os dados de métricas servirem a outro propósito que não a correlação de atividades. Entretanto algumas modificações são necessárias no tipo e na forma de registros de atividades para ser possível utilizar essa fonte de dados, tanto para a análise de métricas quanto para análise de processos. Uma dificuldade encontrada no estudo foi estabelecer o escopo adequado de cada atividade. No caso do projeto P1, as atividades foram atribuídas ao contexto de versão ou OS. Mas os resultados permitiram concluir que, embora adotando um desses dois contextos como unidade de iteração, um critério de classificação ainda pode ser necessário para as atividades dentro de uma única instância. No caso de projetos de software, torna-se claro que o critério de seleção ideal seria analisar a instância sob a visão de fases de projeto. Isto porque todos os modelos de desenvolvimento dividem o ciclo de vida de projetos em fases. Dessa forma cada fase possui seu workflow específico, que pode ser utilizado como parâmetro para comparação.

- Tipo de projeto: desenvolvimento x manutenção

  O tipo de projeto analisado é também uma questão importante a ser levada em conta. Projetos de desenvolvimento e manutenção possuem perfis distintos quanto ao seus processos. Sob a visão de análise de processos, cada um possui pontos positivos e negativos. Pelos motivos já citados, o estudo foi realizado em um projeto de manutenção. Em casos de projetos de manutenção que adotam a estrutura de ordens de serviço, a principal vantagem observada é quanto ao número de instâncias. Como OS abrangem um volume relativamente pequeno de trabalho, muitas dessas iterações são produzidas durante o ciclo de vida do projeto. Com isso se tem um maior número de amostras e em menor granularidade para se analisar. Padrões de desvios podem ser mais facilmente identificados em menor tempo. Porém, também existe uma desvantagem. Pro-

jetos de manutenção são tipicamente voltados a realização de melhorias e correção de defeitos, conforme solicitação do cliente. A realização de melhorias costuma ter um processo melhor definido, podendo ser comparada a um pequeno projeto de desenvolvimento. Entretanto, no caso da correção de defeitos a análise se torna mais complicada. Isto porque muitos eventos são imprevisíveis. Pode-se dizer que o processo para solução de um defeito é menos disciplinado do que o de melhoria, onde é possível fazer um planejamento adequado. Projetos de desenvolvimento são melhor comportados nesse aspecto. De forma que é mais fácil mapear as atividades executadas para o modelo definido. A desvantagem aqui é quanto ao número de instâncias. Modelos de desenvolvimento iterativos como o espiral, evolutivo e processo unificado costumam adotar o conceito de versões de produto. Estas versões representam ciclos de evolução do software ao longo de seu desenvolvimento. No entanto, versões costumam ser produzidas em menor número e com maior duração. Com isso, ao se utilizar versões como instâncias na mineração de processos, o tempo necessário para se obter um volume de dados adequado pode ser muito grande. Para ter a visão de uma única versão completa, por exemplo, poderiam ser necessários meses até que a versão fosse terminada para se obter então o registro de todas as atividades.

- Preparação de dados

  Uma questão também constatada no estudo foi a importância da etapa de preparação dos dados. A etapa exigiu um tempo considerável no trabalho. Além disso, a forma de preparação acabou se revelando um aspecto fundamental na obtenção dos resultados. Principalmente nas tarefas de limpeza dos dados e análise dos mesmos, buscando as melhores formas de estruturá-los para obter resultados mais relevantes. Deve-se ponderar, contudo, a possibilidade desse cenário ser particular para os registros da empresa parceira. A tarefa pode ser mais ou menos fácil dependendo da estrutura dos registros da empresa. No entanto, como em qualquer aplicação de mineração de dados, essa é uma etapa fundamental no processo. A granularidade e qualidade dos resultados dependerá diretamente de uma preparação de dados adequada.

## 4.3 Análise de conformidade: soluções

O presente trabalho estuda o problema de verificar conformidade em PDS. Como parte da solução, a mineração de processos é explorada como ferramenta. A seção atual apresenta duas soluções encontradas na literatura, relacionadas à análise formal de conformidade aplicando mineração de processos. Na seção 4.4 é apresentado um cenário de aplicação, baseado em um dos algoritmos apresentados. Este cenário tem o objetivo de

explorar de forma prática o procedimento envolvido na análise e os resultados fornecidos pela solução.

### 4.3.1 Algoritmos

Pesquisando a literatura se identificou várias soluções de verificação de processos tomando um modelo de execução e um fluxo de eventos. Entre as aplicações estão análise de comunicação de *web services* (10), detecção de violações de segurança em processos (6) e verificação do comportamento de sistemas (16). Contudo, os trabalhos de Cook e Wolf (17) e Rozinat e Aalst (42) se mostraram interessantes para o contexto. Primeiro porque seus requisitos relacionados ao modelo formal e dados de execução serem bastante simples, sendo facilmente satisfeitos no ambiente de software estudado. Também por possuirem métricas definidas para a avaliação da conformidade entre um processo e registros de execução. As duas soluções citadas são detalhadas a seguir.

#### Software Process Validation

O trabalho de Cook e Wolf (17) é pioneiro na análise de conformidade em mineração de processos. Os autores introduzem o conceito de *software process validation*, pelo fato de sua solução ser aplicada a processos de software. São levantadas algumas questões importantes sobre as características de processos de software no trabalho. No entanto, a solução efetiva não é específica para o contexto. Ao final, modelo e execução são considerados como simples fluxos de eventos para fins de aplicação do algoritmo. São propostas duas métricas de *fitness* baseadas na comparação entre dois fluxos de eventos: SSD (*Simple String Distance*) e NSD (*Nonlinear String Distance*). As métricas são calculadas a partir da diferença de distância entre *strings*. Ambos, eventos e modelo, são expressos como cadeias de caracteres. Então as métricas são obtidas com base no número de inserções e deleções de caracteres para obter a equivalência entre as duas *strings*. A métrica SSD calcula o número mínimo de operações necessárias para transformar uma *string* em outra. Ainda são adicionados pesos a cada operação, de forma que é possível atribuir custos a cada uma. A equação 4.1 mostra o cálculo dessa métrica. $W_I$ e $N_I$ representam respectivamente o peso da operação de inserção e o número de inserções. Da mesma forma, na segunda parte do numerador são calculados os custos para operações de deleção. No denominador, $W_{max}$ é o valor máximo entre $W_I$ e $N_I$; e $L_E$ é o tamanho da cadeia de caracteres. A divisão na equação normaliza o valor da métrica entre 0 e 1, independente do tamanho da *string*.

$$SSD = \frac{W_I N_I + W_D N_D}{W_{max} + L_E} \tag{4.1}$$

A métrica SSD é focada no custo individual de operações de transformação, já que cada operação tem seu peso individual. Contudo, para o contexto de processos, os autores consideram interessante diferenciar operações de inserção ou deleção realizadas em sequência. Consequentemente, um bloco da mesma operação em sequência é considerado um desvio mais sério do que se esse mesmo conjunto estiver disperso no log. A equação 4.2 apresenta o cálculo da métrica NSD onde $N_I^B$ e $N_D^B$ são os números de blocos de inserção e deleção, *b* é o comprimento de um bloco em específico e *f(b)* é uma função de custo aplicada a um comprimento de bloco *b*. Os demais termos são os mesmos da métrica SSD. A definição da função de custo *f(b)* provê um parâmetro de ajuste adicional da métrica NSD.

$$NSD = \frac{\sum_{j=1}^{N_I^B} W_I f(b_j) + \sum_{k=1}^{N_D^B} W_D f(b_k)}{W_{max} L_E} \tag{4.2}$$

As duas métricas apresentadas acima fornecem valores únicos como medida da correspondência. No entanto os autores ainda salientam que, derivadas dessas métricas principais, algumas medidas auxiliares são naturalmente obtidas. São elas:

- o número de eventos que correspondem com o fluxo comparado
- o número de inserções e deleções utilizadas para calcular a métrica
- número e tamanho médio dos blocos de operações
- os locais do modelo onde os desvios ocorrem

O trabalho de Cook e Wolf (17) apresentou um método, até então inovador, para comparar dois fluxos de eventos e mostrar as diferenças quantitativamente. Mais recentemente, Rozinat e Aalst (42) apresentam uma proposta semelhante porém com alguns avanços, principalmente relacionados ao tratamento visual das informações de conformidade. Diferentemente de (17), a proposta em (42) utiliza um modelo em rede de Petri como referência. Isto é uma vantagem já que, além de facilitar o entendimento por meio de uma notação gráfica, não necessita da aplicação de técnicas de seleção de um único fluxo dentro das possibilidades do modelo. O tratamento visual que o algoritmo Conformance Checker da aos resultados da análise é realmente um diferencial da solução. Além do provimento de métricas, as discrepâncias entre o modelo e a execução são destacadas e podem ser vistas tanto sob a perspectiva dos registros de execução quanto do modelo prescrito. A interoperabilidade é considerada também um ponto positivo da proposta de (42). O formato XML padrão de entrada e a ferramenta de conversão facilitam bastante o trabalho de pré-processamento de dados para a mineração de processos. Este algoritmo é apresentado a seguir.

Conformance Checker

O algoritmo Conformance Checker (42) utiliza para o cálculo de *fitness*, uma abordagem semelhante à utilizada em (17). No entanto, ao invés de comparar sequências de caracteres esta solução executa o log em uma rede de Petri, calculando o número de inserções e deleções de *tokens* necessárias para essa execução. Além disso, as métricas auxiliares apresentadas nas duas soluções também são semelhantes.

O algoritmo possibilita detectar as discrepâncias entre um modelo e um log comparando-os diretamente, de forma visual e por meio de métricas. As métricas implementadas por este algoritmo pertencem as duas classes, denominadas *fitness* e *appropriateness*. A análise de *fitness* busca investigar se um modelo permite reproduzir todas as sequências de execução dos eventos do log. Em outras palavras, verifica se os traços de execução do log estão conformes com a descrição do modelo. No entanto, uma rede de Petri pode ser capaz de gerar os caminhos contidos no log e ainda permitir comportamentos adicionais. Nesse caso, talvez o modelo contenha caminhos desnecessários. Para medir esse tipo de situação, é introduzida a dimensão *appropriateness*. Essa métrica é analisada sob a perspectiva do modelo. O objetivo é verificar o quão precisamente o modelo captura o comportamento identificado no log. Dessa forma é possível fazer ajustes no modelo. Entretanto, para o caso da análise de conformidade, a noção de *appropriateness* é de menor relevância. Isto porque não se busca melhoria do processo, e sim verificar se ele é seguido adequadamente. Portanto este trabalho irá explorar somente a visão de *fitness*.

Uma forma utilizada para medir a conformidade entre um modelo e um log (*fitness*) é executar o log no modelo e, de alguma forma, medir as discrepâncias. O algoritmo Conformance Checker utiliza essa abordagem. Os eventos do log são executados no modelo de maneira não blocante, ou seja, no caso de faltar um *token* (transição no modelo não contida no log) ele será criado artificialmente e a execução continua. Os *tokens* criados são contabilizados na métrica de *fitness* ao final da execução. A execução inicia colocando um *token* no estado inicial da rede de Petri. Então, os eventos contidos no log disparam transições no modelo. Durante a execução alguns *tokens* terão de ser criados artificialmente (para o caso de a transição presente no log não estar habilitada no modelo e dessa forma não poder ser executada) e outros não serão consumidos (transições habilitadas no modelo que não possuem um evento correspondente no log). A métrica de *fitness* é calculada com base no balanço final entre *tokens* criados e restantes no modelo ao final da execução. Abaixo é mostrada a equação para o cálculo da métrica. Na equação, *k* é o número de trilhas[1] do log. Para cada trilha *i* ($1 \leq i \leq k$), $n_i$ é o número de instâncias combinadas nessa trilha. Este é um artifício utilizado para simplificação dos cálculos. Como várias instâncias

[1] a palavra trilha é utilizada aqui como sinônimo de *trace*. Significa o conjunto de eventos de uma instância completa no log

podem possuir a mesma sequência de eventos, elas são combinadas em uma única trilha. A quantidade de trilhas agrupadas é dada por $n_i$. A variável $m_i$ representa o número de *tokens* faltantes. São aqueles que tiveram de ser criados artificialmente para continuar a execução. $r_i$ é o número de *tokens* restantes, $c_i$ é o número de *tokens* consumidos e $p_i$, o número de *tokens* produzidos (naturalmente, de acordo com a lógica das redes de Petri) durante a execução da trilha corrente no modelo. A métrica *f* então é dada por:

$$f = \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}) + \frac{1}{2}(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}) \tag{4.3}$$

Note que, para todo *i*, $m_i \leq c_i$ e $r_i \leq p_i$. Então, $0 \leq f \leq 1$. Note também que $c_i$ e $p_i$ não podem ser zero porque durante a execução do log existirá pelo menos um *token* produzido no início e outro consumido no fim.

O algoritmo Conformance Checker provê várias formas de visualizar discrepâncias entre um modelo e registros de execução. A solução encontra-se também disponível como um *plugin* no ProM (26) logo, sua interface pode ser explorada amplamente. Além da métrica de *fitness*, diversas outras funcionalidades permitem um diagnóstico completo sobre execução do log no modelo em rede de Petri. Um ponto importante é que a solução não exige que as atividades no modelo e no log sejam equivalentes, ou seja, possuam o mesmo nome e mesma granularidade. É provida uma funcionalidade de mapeamento entre as duas fontes de dados. Dessa forma, uma pessoa com conhecimento sobre os processos pode identificar visualmente as correspondências entre as atividades do modelo prescrito e os registros de execução. Até mesmo um metamodelo de desenvolvimento padrão pode ser adotado diretamente, sendo feita então a correspondência para os nomes efetivamente registrados para cada atividade. Além disso, uma atividade no modelo pode ser ligada com vários eventos no registro e vice-versa. Isso permite gerenciar diferenças de abstração entre modelo e dados de execução. Por fim, uma atividade no modelo ainda pode ser tornada invisível para o caso de ela nunca ocorrer nos registros. Ela não será considerada durante a análise de conformidade. Essa função também é importante pois permite a um conhecedor do domínio fornecer uma heurística ao algoritmo, evitando assim a detecção incorreta de divergências. Após o mapeamento, os resultados da análise podem ser vistos sob duas perspectivas:

1. **Perspectiva do modelo**

   Com a visão do modelo, a métrica de *fitness* é mostrada. Ela é calculada com base nas relações entre *tokens* produzidos, restantes, consumidos e faltantes, conforme a equação 4.3. Se o log é executado corretamente no modelo sem nenhum *token* faltante ou restante no final, o *fitness* é 1. Há ainda uma série de opções para facilitar a identificação das diferenças. Todas as funcionalidades são explicadas a seguir e

podem ser observadas na figura 4.11. À esquerda na janela estão as instâncias do log, que podem ser selecionadas individualmente. Ao centro é apresentado o modelo com todas informações de inconformidades, podendo ser cada uma delas desabilitada. O valor de *fitness* pode ser visto à direita na figura.

- *Token counter*: Mostra os *tokens* faltantes e restantes para cada transição do modelo. Permite localizar precisamente as partes onde ocorreu um problema.
- *Failed tasks*: Destaca as atividades que não foram habilitadas e consequentemente não sendo executadas. Aparecem em amarelo no modelo.
- *Remaining tasks*: Essas são atividades que deveriam ser executadas (possuem um *token* na entrada) mas não ocorreram no log. São destacadas em cinza no modelo.
- *Path coverage*: É possível visualizar todas as transições executadas, não importando se foram executadas com sucesso ou tiveram de ser forçadas, gerando *tokens* faltantes. Essa funcionalidade permite acompanhar o caminho de uma ou várias instâncias no log. As atividades são coloridas em verde.
- *Passed edges*: Mostra em cada conexão, o número de vezes em que ela foi executada considerando todas as instâncias do log.

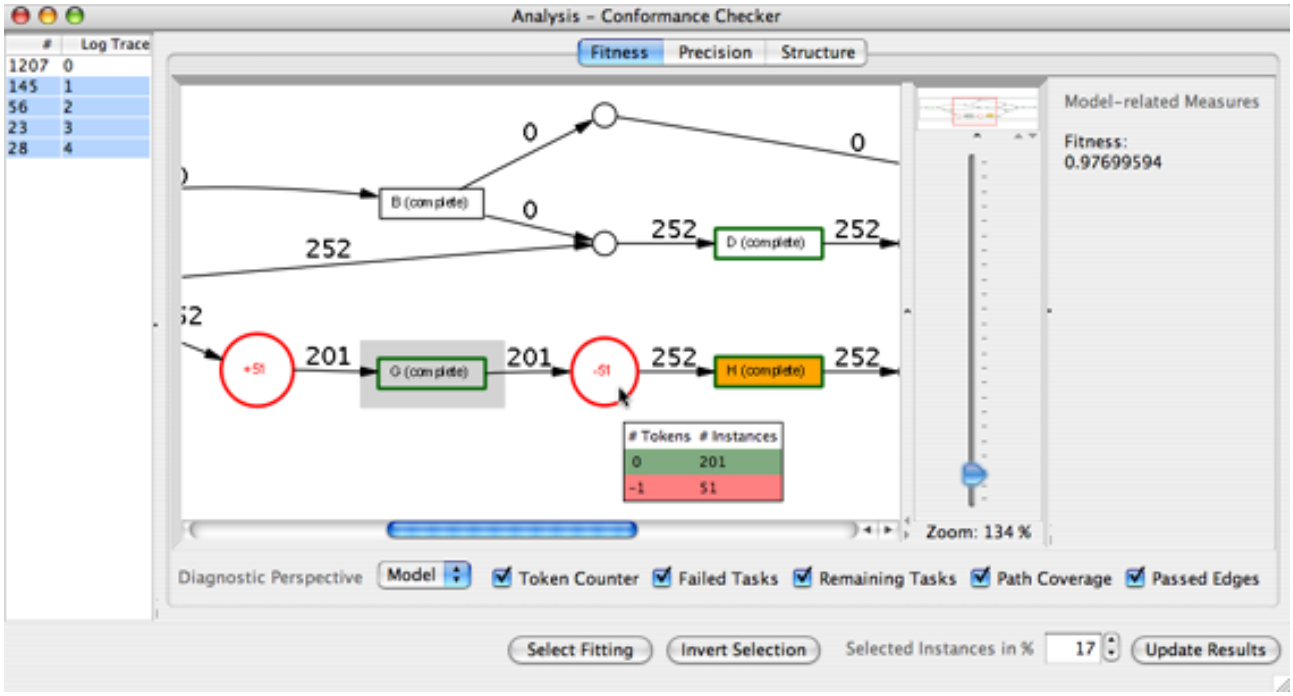


Figura 4.11: Tela de resultados de conformidade do algoritmo: visão do modelo

2. **Perspectiva do log** A perspectiva de diagnóstico pode ser modificada, permitindo analisar para o log ou um subconjunto de suas instâncias. São providas duas métricas

e também os eventos não executados no modelo são destacados em amarelo, como pode ser visto na figura 4.12 (Failed log events). Para cada evento apresentado nessa perspectiva também é indicado o responsável por sua execução. A informação retirada do campo *originator* do log MXML, permite identificar o recurso envolvido em cada atividade de cada instância específica.

- *Successful execution*: Métrica dada pela fração instâncias de processo executadas com sucesso, levando em conta o número de ocorrências por trilha.
- *Proper completion*: Métrica dada pela fração instâncias de processo completadas propriamente, levando em conta o número de ocorrências por trilha.
- *Failed Log Events*: Destaca na visão do log os eventos que não puderam ser executados corretamente.

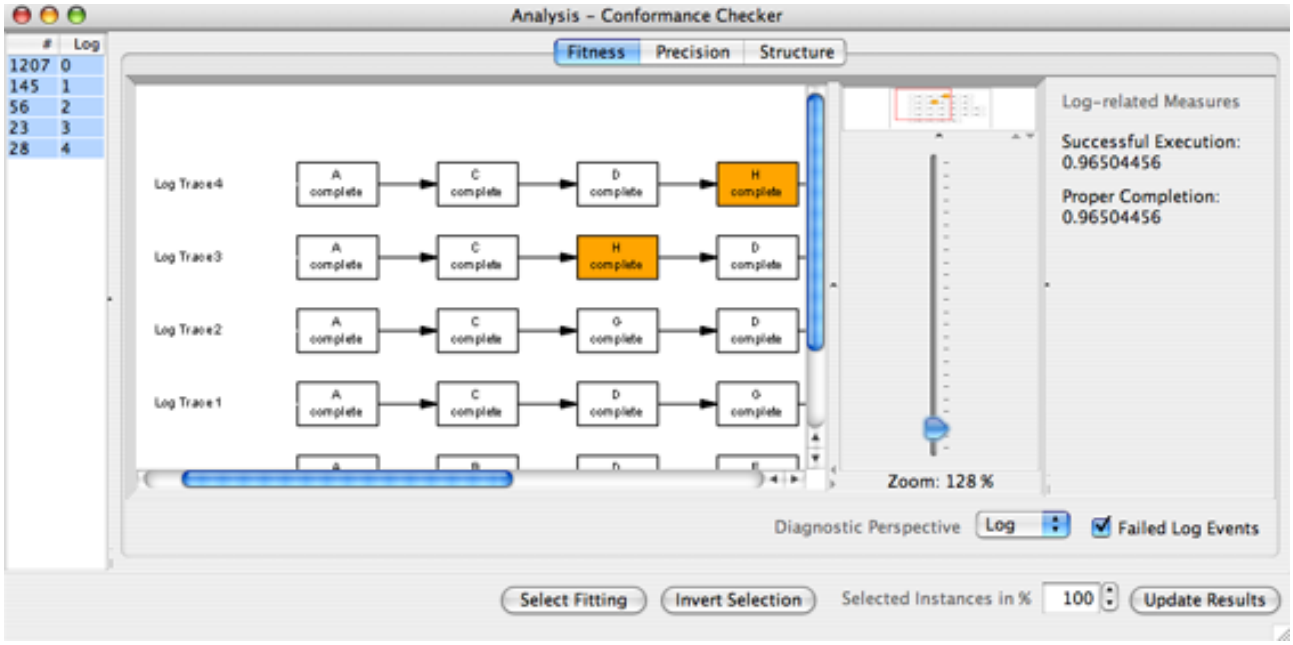


Figura 4.12: Tela de resultados do algoritmo: visão do log

## 4.4 Análise de conformidade: aplicação

Enfim, a partir do estudo descrito conclui-se que técnicas de mineração de processos podem ser sim aplicáveis no contexto de PDS, levando em conta suas características específicas. Esta seção apresenta uma série de considerações acerca da análise de conformidade em PDS, aplicando mineração de processos. A discussão é feita fundamentalmente com base no ambiente encontrado na empresa parceira. Uma aplicação prática foi realizada com o algoritmo Conformance Checker (42), utilizando os mesmos dados do projeto P1 para a fase de testes.

### 4.4.1 Modelo predefinido

O primeiro passo para comparar um processo definido com sua execução é obter um modelo desse processo. Para a avaliação com mineração de processos, normalmente é necessário um modelo em rede de Petri. Tal modelo poderia ser obtido de três diferentes maneiras: (i) inferir utilizando técnicas de descoberta de processos por mineração, utilizando os próprios registros de execução, (ii) adotar um modelo construído pela própria empresa ou (iii) utilizar um modelo padrão de desenvolvimento como por exemplo, Processo Unificado. Cada uma dessa alternativas possui vantagens e desvantagens. A adoção de uma delas dependerá de fatores específicos de cada empresa. A primeira abordagem pode facilitar o mapeamento entre modelo e dados de execução. Além disso, o modelo produzido pela descoberta a partir do histórico de execução provavelmente será mais próximo da realidade do projeto. No entanto, o estudo descrito na seção 4.2 mostrou que descobrir processos a partir de registros de PDS pode apresentar algumas dificuldades. Primeiramente devido aos processos de software serem normalmente pouco estruturados. Em segundo, algoritmos de mineração de processos exigem um volume grande de instâncias para descobrir um modelo preciso, o que não é uma situação comum no ambiente de desenvolvimento de software. Por fim, mesmo que represente a realidade, o modelo produzido pode não ser considerado satisfatório pela empresa. Nesse caso, a melhor alternativa é adotar um modelo previamente construído como padrão. Dessa forma a organização pode ter uma visão clara sobre a semelhança entre o fluxo ideal/esperado para o processo e a realidade representada pelos dados de execução. A transformação adequada do modelo para uma rede de Petri possibilita a avaliação formal utilizando algum dos algoritmos apresentados anteriormente ou mesmo outras soluções. Por último, um metamodelo de processo pode ser utilizado. Metamodelos de PDS costumam expressar detalhadamente o workflow de cada fase de projeto. O procedimento é semelhante ao da alternativa anterior. A única condição em qualquer um desses dois casos é que seja possível fazer um mapeamento (mesmas atividades/mesmo nível de abstração) entre o modelo e os dados de execução do processo.

#### Exame de um caso real

Com relação ao modelo de processo utilizado para comparação foram consideradas duas alternativas: adotar os modelos documentados no OSSP da empresa parceira ou obter por mineração. As duas hipóteses foram exploradas com o intuito de identificar as diferenças. Na primeira abordagem foi utilizada uma ferramenta de desenho de redes Petri para construir o modelo, baseado no workflow de testes da organização. Esse workflow foi construído baseado estritamente na documentação disponível. A única modificação feita foi a separação das atividades em início e fim. Isto foi feito observando a estrutura dos dados

reais de execução. Note-se que uma mesma atividade na base de métricas pode estar registrada em várias linhas, dependendo do número de vezes em que o colaborador trabalhou na tarefa. Para tornar os dados coerentes com a representação de processos, todas as atividades foram divididas nas etapas de inicio e fim. Entretanto, os modelos de workflow documentados não prevêem esse tipo de situação. Sendo assim, mesmo que o processo seja executado na ordem correta o algoritmo irá detectar inconformidade quando houver repetição de atividades. Para possibilitar a comparação adequada, levando em conta a duração de cada atividade, o modelo também deve conter explicitamente identificadores para o início e fim dessas atividades. Embora essa modificação já possibilite a verificação, o modelo ainda continua sequencial. O processo em rede de Petri foi derivado dos modelos de workflow originais do OSSP da empresa, que por sua vez não expressam paralelismo. Consequentemente, atividades executadas em paralelo serão interpretadas como inconformes com o processo definido. Nesse caso é importante que a organização identifique a concorrência entre atividades, seja por algum conhecimento prévio sobre seus projetos ou mesmo com o auxílio da mineração de processos. Os resultados do estudo exploratório mostram que esta última abordagem pode ser inclusive um bom meio de compreender a realidade dos processos, visualizando detalhes como concorrência, pontos de decisão e loops.

### 4.4.2 Dados de execução

Os dados utilizados nessa aplicação foram essencialmente os mesmos do experimento relatado no estudo exploratório. A análise foi aplicada sobre o processo de testes, adotando a versão de produto como instância. 21 versões foram selecionadas, sendo que nenhum procedimento adicional de preparação de dados foi feito além daquele já descrito na seção 4.2.3. Esta é inclusive uma estratégia interessante, já que permite explorar os dados de duas formas. Primeiro, os registros podem ser submetidos a um algoritmo de descoberta de processos. Os modelos obtidos podem ser analisados visualmente, o que já fornece algumas pistas sobre o comportamento do projeto. Indicadores fornecidos por esses algoritmos também podem ser explorados. Adicionalmente pode se utilizar um algoritmo de análise de conformidade como os apresentados na seção 4.3.1. Os resultados desse tipo de solução podem complementar a visão do auditor sobre os processos executados com indicações e medidas objetivas sobre a conformidade.

### 4.4.3 Execução do algoritmo e resultados providos

Tendo os dados de execução devidamente preparados e pré-selecionados, é possível então aplicar um algoritmo de análise de conformidade para analisar o processo. Note-se que o presente cenário é baseado no algoritmo Conformance Checker. Antes de executar

o algoritmo, cada atividade do modelo definido deve ser relacionada com um evento nos registros de execução. Como o algoritmo utiliza um modelo em rede de Petri para comparação, cada evento do log irá disparar a transição correspondente no modelo (se existir), de acordo com o mapeamento realizado. Este algoritmo fornece uma funcionalidade para tal mapeamento, como pode ser visualizado na figura 4.13. Na coluna esquerda da janela são mostradas as atividades do modelo predefinido. No centro, encontram-se as atividades que aparecem pelo menos uma vez no log. A coluna da direita mostra a identificação final que aparecerá no modelo, resultante do mapeamento. Nessa etapa espera-se que alguém com conhecimento, tanto sobre os processos quanto sobre os dados, seja capaz de identificar a correspondência entre as atividades. Note-se que eventos que não aparecem nos registros não poderão ser mapeados para o modelo. Nesse caso, o evento do log pode ser tornado *visível* ou *invisível*. No primeiro caso a atividade será inserida explicitamente no modelo, como pode ser observado na atividade *Design* da figura 4.13, por exemplo. Entretanto, como a atividade não aparece no registro, isso poderá resultar em um ponto de inconformidade caso a execução de tal atividade seja obrigatória no processo predefinido.

Alternativamente, uma atividade pode ser tornada invisível. Eventos invisíveis nas redes de Petri são geralmente utilizados para controle, não fazendo parte do processo em si. Nesse caso ficará por conta do algoritmo disparar automaticamente essas transições durante a avaliação do log, de acordo com alguma heurística. A heurística utilizada pelo Conformance Checker é a do caminho mais curto. Isto significa que, na presença de mais de um caminho com atividades invisíveis, será disparada a transição de menor caminho. Tal funcionalidade é útil quando se sabe previamente que uma atividade do modelo nunca acontece na execução, evitando então a geração indevida de uma inconformidade. Após o mapeamento das atividades o algoritmo pode então ser executado. Um último ponto a ser observado antes da análise diz respeito à busca de atividades invisíveis. É possível restringir o número de atividades buscadas. Neste cenário o nível de busca foi estabelecido em 0 porque nenhuma atividade invisível está presente no modelo. Considerando para comparação os modelos definido e minerado, esses dois cenários são discutidos a seguir.

### Comparando com o modelo de processo organizacional

Em um primeiro cenário, é possível comparar a execução diretamente com o modelo de processo documentado pela empresa. Na figura 4.14 é apresentado o painel de resultados de análise do algoritmo Conformance Checker, para o processo de testes da empresa parceira. São fornecidas informações detalhadas sobre a execução do processo e as discrepâncias encontradas. O auditor aqui pode visualizar a conformidade do processo, tanto de forma geral quanto individualmente, selecionando uma instância em específico. Além disso a análise pode ser feita sob a perspectiva do modelo ou do log. A figura 4.14 mostra

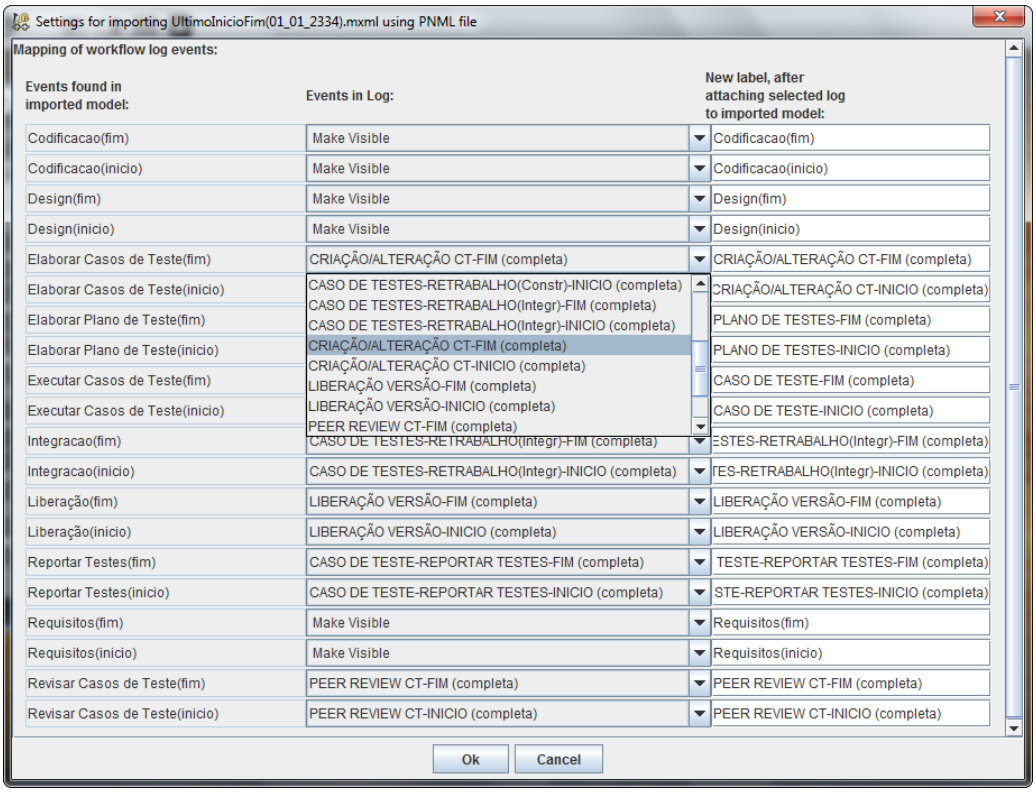


Figura 4.13: Mapeamento de atividades entre modelo e registros

os resultados vistos sob a perspectiva do modelo. Com todas as instâncias (versões) selecionadas é mostrado o valor médio de *fitness*, que para os dados do projeto P1 ficou em torno de 0,64. Essa métrica já provê uma noção sobre a conformidade global do processo. Quanto maior for o número de instâncias com atividades não executadas ou executadas fora de ordem, menor será o valor de *fitness*. Para servir como parâmetro, foi criada artificialmente a versão 00.00.00. É simulada a execução perfeita de uma iteração do processo, passando uma vez por cada atividade até a liberação da versão. Como se pode observar na figura 4.15, a instância é 100% conforme ( *fitness*=1).

Mesmo fornecendo uma noção quantitativa sobre a conformidade, a medida de *fitness* é uma informação restrita, baseada em um único valor. Mas, outras informações disponibilizadas pelo algoritmo podem ser exploradas. O número de vezes que cada transição pode ser visualizado com o indicador *Passed Edges* habilitado. Dessa forma é possível identificar precisamente as atividades e caminhos mais frequentes, o que também é uma pista importante sobre os padrões de execução (correta ou incorreta) do processo.

Ainda sob a visão do modelo, as cores utilizadas para enfatizar certos aspectos também auxiliam na análise. Na figura 4.14 se observa eventos marcados com três cores. Em verde são destacados aqueles que ocorreram pelo menos uma vez no log. Em amarelo, os eventos que foram executados com sucesso por não estarem prontas (sem *token* na entrada). Isto indica normalmente atividades executadas fora de ordem. São mostrados em cinza os

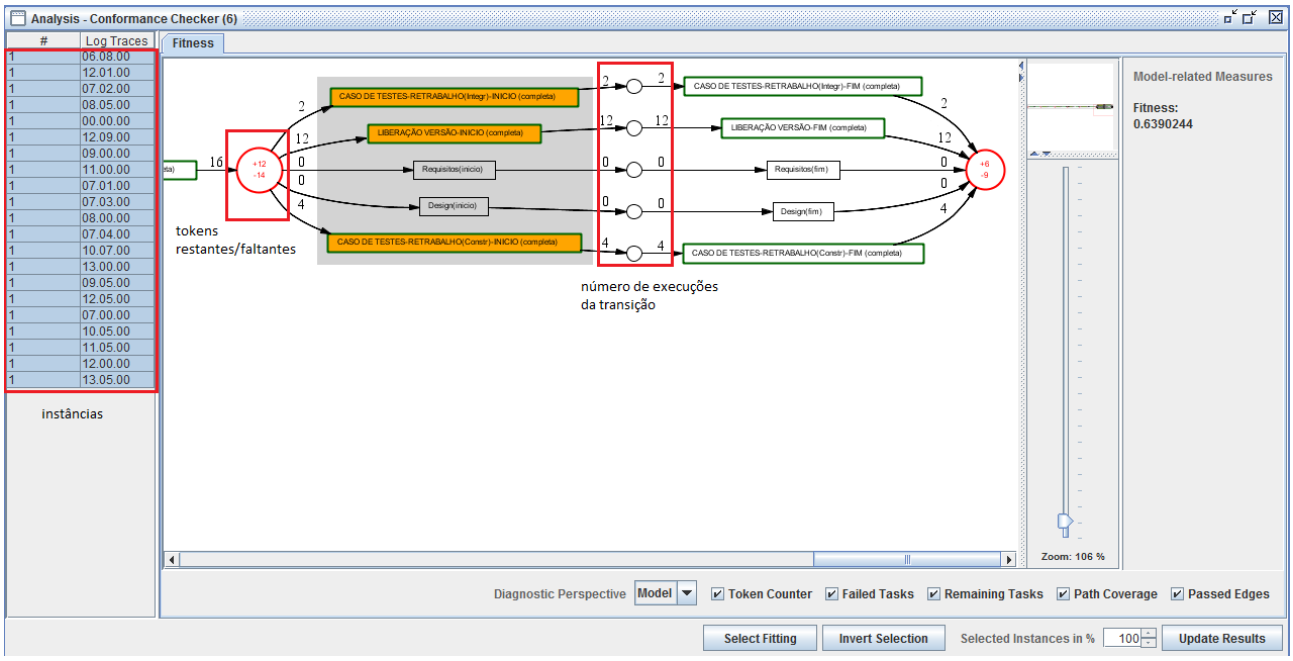


Figura 4.14: Tela de resultados do algoritmo

eventos que deveriam ocorrer (um *token* na entrada) mas não apareceram no log. Além disso, o processo pode ser visualizado sob a perspectiva de execução. Tal funcionalidade é interessante, visto que ela permite observar a sequência exata e o número de atividades que foram efetivamente executadas em cada instância. Observe na figura 4.16 que a versão 07.01.00, por exemplo, possui apenas duas atividades executadas e ainda fora da ordem esperada. Isto explica seu baixo valor de *fitness*, aproximadamente 0,4. A informação sobre os recursos envolvidos, contida em cada atividade, também pode ser útil. Na figura se observa que as duas atividades foram realizadas pelo mesmo colaborador. No caso de desvios muito frequentes originados pelo mesmo responsável, por exemplo, o agente de SQA poderia interferir pontualmente junto a esse colaborador buscando identificar a causa do desvio.

### Comparando com o modelo minerado

O cenário anterior é baseado em um modelo padrão definido pela organização. Contudo, pode ser desejável comparar a execução com um modelo inferido por mineração. Considerando os aspectos discutidos na seção 4.4.1, um modelo de processo obtido por mineração pode ser uma alternativa interessante visto que se tem um fluxo mais próximo da realidade do projeto. A figura A.2 mostra um modelo em rede de Petri produzido pelo algoritmo Heuristics Miner, descrito na seção 4.2. Embora o modelo original esteja em uma notação própria, a ferramenta ProM disponibiliza um *plugin* que permite fazer a transformação diretamente dessa notação para uma rede de Petri.

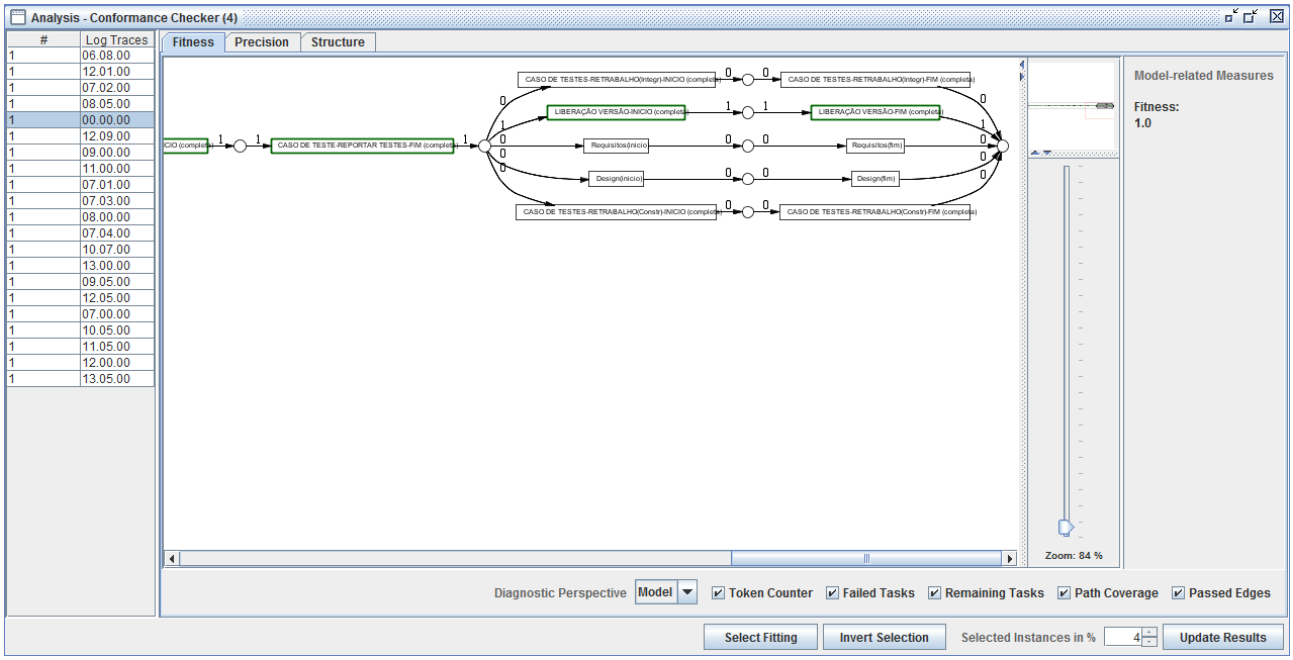


Figura 4.15: Versão 00.00.00 selecionada - 100% conforme

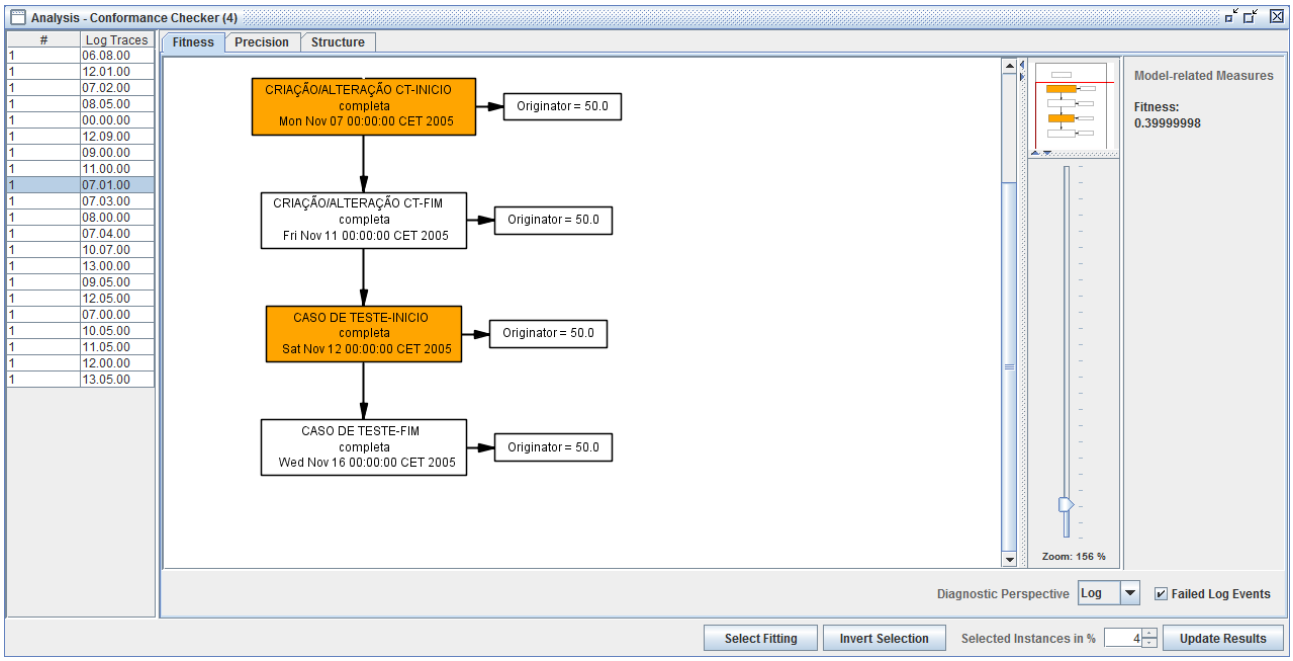


Figura 4.16: Sequência de atividades da versão 07.01.00 - perspectiva do log

De fato esse modelo quando comparado com a execução do processo apresenta maior semelhança. Isto pode ser verificado pelo valor de *fitness*, que foi maior do que o obtido pela comparação com o modelo do OSSP. Seu valor foi de 0,81. Se observa, contudo, que o fluxo do processo minerado possui algumas diferenças com relação ao definido na documentação organizacional. A existência de tal discrepância é também uma questão importante a ser analisada, onde podem ser consideradas duas hipóteses: ou o processo apresenta inconformidades bastante frequentes, ou na verdade o modelo definido não representa adequadamente o comportamento normal dos seus processos. Cabe ao grupo de qualidade responder tal questão e decidir por alterar o processo definido ou interferir juntamente aos projetos, buscando eliminar tais inconformidades.

## 4.5 Considerações sobre o capítulo

Se constatou no estudo realizado, que a mineração de processos pode trazer benefícios na tarefa de analisar a execução de processos de software. As diferenças entre os processos definidos e executados podem ser acessadas, tanto na visualização dos modelos produzidos por algoritmos de descoberta de processos, quanto pela avaliação de métricas fornecidas por algoritmos de análise de conformidade. O estudo permitiu traçar parte do cenário que pode envolver a verificação de conformidade em PDS, utilizando técnicas de mineração de processos como ferramenta. Portanto, derivado da experiência obtida foram compiladas algumas sugestões para a aplicação de uma solução desse tipo no ambiente de desenvolvimento de software. Embora várias dessas questões já tenham sido levantadas anteriormente, elas são sumarizadas a seguir.

### 4.5.1 Sugestões ao analisar conformidade em PDS aplicando mineração de processos

Ao verificar conformidade em PDS, possuindo um ambiente semelhante ao descrito no presente trabalho, certas questões devem ser levadas em conta.

- Processos

    1. Verificar a existência de um modelo de workflow do(s) processo(s) analisado(s). Se existir, deve ser possível mapear cada atividade para correspondente nos dados de execução. Além disso, o modelo deve ser transformado para uma rede de Petri possibilitando a aplicação direta em um algoritmo de análise de conformidade. Não havendo um modelo documentado, ainda é possível inferir por mineração. No entanto, questões como dados insuficientes, incompletos e/ou incorretos podem reduzir a qualidade dos modelos produzidos.
    2. Identificar o nível de abstração dos modelos documentados: para a obtenção de resultados mais relevantes, é ideal que o modelo de processo contenha pelo

menos as estruturas básicas de fluxo de controle modeladas (sequência, concorrência, laços e decisões).

3. Validar os parâmetros do algoritmo de descoberta de processos utilizado (se a inferência de um modelo for considerada): algoritmos normalmente possuem parâmetros/heurísticas a serem definidos anteriormente à mineração. A realização de um procedimento formal de identificação dos valores de parâmetros mais adequados acrescenta maior confiabilidade aos resultados produzidos. Uma possível abordagem poderia envolver a execução iterativa do algoritmo buscando valores limites. Diversas situações podem ser tratadas como limite. A mudança de relações entre atividades e a inclusão/exclusão de atividades no modelo resultante são dois exemplos.

- Dados

1. Verificar se os dados atendem aos requisitos mínimos para mineração: identificador de instância, atividade, início e fim de cada atividade (ano, mês, dia e hora) são requisitos mínimos para análise de fluxo de controle. No caso dos algoritmos apresentados, que não utilizam o campo *EventType*, a diferenciação entre início e fim de atividade deve ser feita junto ao identificador da própria atividade, conforme pode ser observado na figura 4.8. Adicionalmente, a inclusão do colaborador envolvido na tarefa permite estender a análise do processo fornecendo informações sob a perspectiva organizacional. É possível, por exemplo, verificar se a correspondência entre atividade/papel prevista no modelo é realmente obedecida.
2. Tratar atividades duplicadas: no caso da empresa parceira, o registro de atividades é feito diariamente pelos colaboradores. No fim do dia, cada um registra as atividades em que esteve envolvido e as horas investidas em cada uma. Tipicamente uma atividade pode ter duração de vários dias. Consequentemente, essa atividade irá conter múltiplas entradas no registro. É importante que tal situação seja tratada antes de submeter os dados para mineração. Sob a visão de eventos de processo, atividades repetidas são identificadas como um laço. Entretanto, nos modelos de workflow de PDS analisados não é expresso tal comportamento. O tratamento realizado no cenário de aplicação apresentado consistiu em utilizar a primeira e última entradas da mesma atividade, criando um registro da forma início/fim e eliminando as atividades repetidas. Porém, essa abordagem pode acarretar em perda de informação. Isto porque não é feita distinção entre uma única tarefa, que é registrada em várias partes e um laço efetivamente. Essa distinção não foi feita no estudo exploratório por não ser possível capturar a infor-

mação necessária para tal. Como exemplo, vários registros de um plano de teste podem ser abstraídos em uma única atividade com duração determinada. No entanto, caso de teste pode ser realmente uma atividade iterativa, sendo executado uma vez para cada caso de teste previsto no plano. Nesse caso, a atividade poderia ser mantida no registro, sendo necessário, contudo, explicitar tal comportamento no modelo predefinido.

3. Avaliar a necessidade de um procedimento de limpeza nos dados: esse item se aplica a qualquer contexto que envolva extração de conhecimento sobre conjuntos de dados. Portanto, a qualidade dos dados deve ser uma questão a ser analisada criteriosamente, também envolvendo a mineração de processos. Em se tratando de registro de atividades, quanto menor o número de restrições impostas aos colaboradores durante a inserção desses dados, maior terá de ser o trabalho necessário na verificação e posterior eliminação de dados indesejados. Nos registros da empresa parceira, a implementação de algumas restrições simples melhoraria sensivelmente a qualidade dos dados, por exemplo: não permitir campos de data em branco, impor um formato padrão para identificação da OS/versão relacionada (se encontrava, por exemplo, OS registradas da forma: OS-XX, OSXX, XX, etc.), definir algumas diretivas para a inserção de comentários sobre a tarefa realizada. Essas são algumas das restrições que poderiam diminuir o volume de dados descartados, consequentemente melhorando a qualidade dos achados.

- Resultados

1. Considerar outros indicadores: para uma visão mais consistente sobre a execução do processo, outros indicadores podem ser explorados. No algoritmo Conformance Checker, por exemplo, as atividades não executadas ou executadas fora de ordem são destacadas visualmente, de forma que podem ser facilmente identificadas, contabilizadas, servindo como mais um indicador de conformidade. Além disso, métricas fornecidas pelos algoritmos de descoberta de modelos também podem ser exploradas. Os valores de significância das atividades e correlação entre elas podem fornecer pistas sobre o fluxo mais frequente de execução de um processo.
2. no algoritmo Conformance Checker, explorar as visões do modelo e do log: sob a perspectiva do modelo, o algoritmo apresenta informações como a medida de *fitness*, atividades não executadas e número de execuções de cada transição. Contudo, notou-se que em certos casos a análise sob essa perspectiva pode dificultar a visualização correta da conformidade. Em certas situações, um *token*

disponível permite que uma atividade seja executada corretamente, mesmo que fora de ordem. Por exemplo, se a primeira atividade do processo aparecer em qualquer ponto do registro, ela será mostrada como conforme no modelo. Isto ocorre porque o algoritmo sempre inicia com um *token* no início da rede de Petri. Outras combinações também podem gerar situações semelhantes, distorcendo os resultados. Portanto, é interessante que a perspectiva do log também seja acessada, permitindo-se visualizar a ordem efetiva na qual as atividades foram executadas (ver figura 4.16). Atividades não executadas também podem ser facilmente identificadas, visualizando o processo sob essa perspectiva.

# 5. Trabalhos relacionados

A busca por formas de monitorar e melhorar processos de software não é nova. São encontradas na bibliografia diversas contribuições tratando do assunto. Este capítulo apresenta os trabalhos vistos como mais relevantes para a pesquisa, envolvendo a exploração de dados de execução de PDS para descoberta de conhecimento sobre processos e também análise de conformidade.

## 5.1 Sorumgard [SOR97]

Sorumgard (46) estudou verificação de conformidade na engenharia de software, de maneira experimental. Nesse sentido, as contribuições do trabalho se concentram em quatro aspectos principais:

- Definir um modelo de conformidade para a engenharia de software
- Estabelecer um guia para modificação de processos, de forma que ele possa permitir a medição da conformidade em sua execução
- Estabelecer um guia para modificação de processos, de forma que ele possa permitir a medição da conformidade em sua execução
- Testar a proposta por meio de um experimento controlado

Com relação à definição do modelo, é introduzido o conceito de *vetor de desvio*. Esse vetor serve para representar a conformidade de acordo com determinadas dimensões. Dimensões são interpretadas aqui como propriedades ou elementos de um processo. O modelo de conformidade proposto é definido com base em parâmetros, que pertencem a três categorias: processos, produtos e recursos. Para cada um desses parâmetros são estabelecidos atributos observáveis. Por exemplo, um processo possui tempo, esforço, número de eventos e custo. Um produto possui uma medida de tamanho e qualidade. Assume-se que, para cada um desses atributos, possam ser definidos valores esperados. Tais medidas comparadas com as obtidas na execução do processo produzirão um vetor de desvio, conforme ilustrado na figura 5.1. Esse vetor de desvio pode ter um número qualquer de dimensões, dependendo dos atributos observados. A figura 5.1 mostra um possível vetor envolvendo as dimensões *tempo* e *qualidade*. As variáveis $C_{i,p}$ e $Q_{i,p}$ representam os valores esperados (*predicted*) para as dimensões em questão. Já os valores medidos na execução estão representados por $C_{i,e}$ e $Q_{i,e}$. O vetor de desvio expressa então a diferença vetorial entre o processo pressuposto e o realizado. Por exemplo, se o processo é executado exatamente como esperado o vetor de desvio terá valor [0,0].

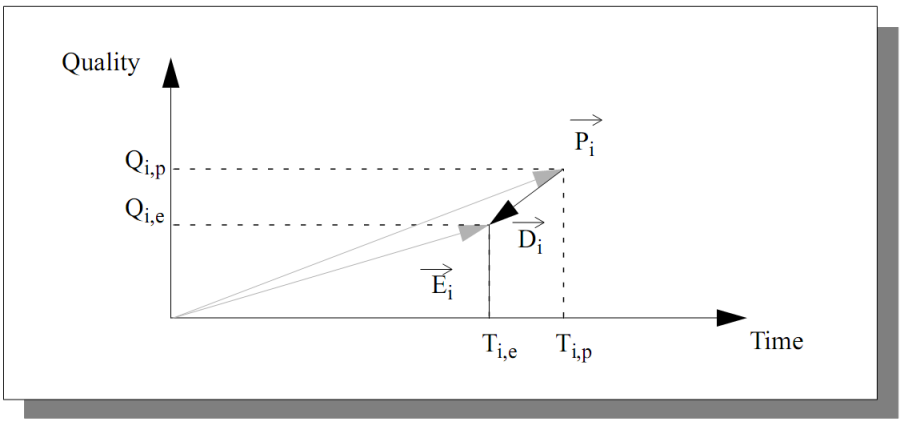


Figura 5.1: Vetor de desvio que representa a conformidade. Fonte:(46)

O trabalho é bastante consistente, abrangendo vários aspectos de um processo relacionados à discrepância entre planejamento e execução. No entanto, é focado na análise da diferença entre medições obtidas sobre cada atributo envolvido no processo. O trabalho assume que, uma vez definidos os atributos necessários para medir a conformidade, existem valores pré-definidos para cada dimensão. Dessa forma o trabalho foca em definir formalismos matemáticos para obter o vetor de desvio que representa a conformidade do processo.

## 5.2 Huo et al. [HZJ06]

Neste trabalho os autores propõem um método para a descoberta de padrões sobre dados de PDS. Tais padrões são expressos em um modelo em rede de Petri que, segundo os autores, pode ser usado para ser comparado com um modelo padrão de desenvolvimento adotado pela empresa. Para descoberta de relações entre atividades, o trabalho propõe uma solução, baseada no algoritmo $\alpha$ (8). Entretanto, o algoritmo proposto em (32) busca superar as limitações existentes naquela solução. Primeiro, o algoritmo $\alpha$ tenta descobrir um modelo completo. Consequentemente, o conjunto de dados precisa ser completo. A solução proposta busca inferir qualquer padrão recorrente identificado nos dados, não necessitando de instâncias do log com todas as atividades executadas. Essa abordagem se aproxima da utilizada por métodos estatísticos (50) (27). Porém, os autores argumentam que tais soluções exigem um grande volume de dados, o que não é uma situação comum em projetos de software. Além disso, os autores salientam a necessidade de garantir a equivalência no nível de abstração entre os padrões obtidos por mineração e o modelo predefinido. No contexto apresentado, as atividades registradas podem estar em menor granularidade do que os elementos do modelo. Para resolver o problema os autores sugerem o mapeamento manual, feito por pessoas conhecedoras do domínio. Para garantir um mapeamento correto é utilizado o método Kappa, para medir a concordância entre difer-

entes avaliadores. Dessa forma é garantido que o modelo/padrão minerado poderá ser comparado diretamente com o modelo de desenvolvimento predefinido.

Uma restrição identificada neste trabalho, contudo, é relacionada à forma da comparação pressuposta. Primeiro, não é considerada a análise de instâncias em separado, já que os padrões inferidos são obtidos de todo o conjunto de dados. Segundo, a comparação se daria apenas de forma visual, sem a produção de nenhuma medida objetiva sobre a conformidade entre modelo e execução. No entanto, a contribuição é vista como complementar à pesquisa por tratar um problema inclusive já identificado no presente trabalho, que envolve o mapeamento e seleção correta das atividades na base de métricas para realização da comparação de forma adequada.

## 5.3 Silva e Travassos [ST04]

Assim como em (46), Silva e Travassos (45) também estudaram a verificação de conformidade na engenharia de software experimental. Essa área é identificada pelos autores como carente desse tipo de verificação. São citados estudos constatando que normalmente pessoas não seguem os processos propostos durante a realização de experimentos. O objetivo principal do trabalho trata-se da implementação e avaliação (por meio de experimentos com observação não intrusiva) de uma ferramenta para auxílio na inspeção de artefatos em PDS. A ferramenta é baseada no conceito Perspective Based Reading (PBR) (45) para inspeção. Seus objetivos principais são: i) guiar o executor da inspeção de acordo com a técnica proposta e ii) facilitar a descrição das discrepâncias identificadas. Além disso a ferramenta é capaz de coletar dados sobre a execução da inspeção (tempo total de inspeção, tempo e ordem das tarefas na identificação de discrepâncias, número de consultas a uma ferramenta de ajuda, etc.). Essa funcionalidade serve para monitorar o utilizador da ferramenta de forma não intrusiva, para garantir que a técnica definida na ferramenta foi realmente seguida. Entretanto, o trabalho cita a utilização das métricas em um experimento realizado para avaliar o uso da própria ferramenta. Embora não contendo nenhum dado conclusivo, os autores esperam responder questões como:

- A eficiência dos revisores é maior quando utilizando a ferramenta?
- O uso da ferramenta pode motivar os revisores a seguirem as técnicas propostas?
- A ferramenta pode melhorar o entendimento sobre a técnica?

Porém, os autores não descrevem detalhes sobre a forma na qual essas métricas deveriam ser interpretadas, buscando avaliar conformidade na execução da técnica PBR. O foco do trabalho se restringe a discutir algumas abordagens de monitoração de execução para analisar conformidade, além da implementação da ferramenta.

Tabela 5.1: Comparação entre algoritmos de mineração de processos

| **Dimensões** | (46) | (45) | (32) | [CRU10] |
|---|---|---|---|---|
| Contexto | PDS | PDS | PDS | PDS |
| Objetivo | Verificar conformidade | Avaliar ferramenta/Verificar conformidade | Inferir padrões recorrentes | Verificar conformidade |
| Dados utilizados | Formulários preenchidos manualmente durante um experimento controlado | Métricas coletadas por uma ferramenta específica durante um experimento controlado | Banco de dados de um EPG | Base de métricas de software |
| Preparação de dados | Não envolve | Não envolve | Não definido | Discutida em um estudo de caso exploratório |

## 5.4 Considerações sobre o Capítulo

Embora o problema da conformidade em PDS seja bastante citado, os trabalhos encontrados não discutem detalhes sobre como uma solução para verificação de conformidade poderia ser implementada nesse ambiente. Os trabalhos analisados não tratam objetivamente da estrutura, fonte e preparação de dados necessárias para viabilizar a exploração de dados de execução em PDS. Além disso, as contribuições encontradas são pontuais. Processos de software não são tratados amplamente, sob todas suas perspectivas incluindo: ordem atividades, recursos e artefatos. Com exceção de (32), as outras contribuições assumem a avaliação da conformidade do processo pela simples coleta de medidas específicas, posteriormente comparadas com valores estimados. No entanto, tanto o fato de apenas coletar métricas, quanto as próprias métricas identificadas nos trabalhos em si, não são vistas como suficientes para se ter uma visão consistente sobre a aderência aos processos na engenharia de software. Tendo isso como fator motivador, o presente trabalho busca melhorar o entendimento sobre o assunto, analisando o problema em um ambiente real, que envolve a operação de software da empresa parceira.

# 6. Conclusão

## 6.1 Considerações Finais

O presente trabalho teve como principal objetivo explorar a mineração de processos como ferramenta auxiliar na verificação de aderência aos processos definidos em um ambiente de desenvolvimento de software. Inicialmente, um estudo de caso exploratório confirmou a viabilidade em descobrir relações entre eventos de um processo típico de software, tendo como fonte de dados uma base de métricas de esforço semelhante àquela encontrada na empresa parceira. As lições aprendidas com esse primeiro estudo permitiram identificar questões importantes a serem consideradas, como:

- Formato típico dos processos definidos em PDS;
- Algoritmos mais adequados para mineração de dados de execução nesse ambiente;
- Escolha adequada dos dados para mineração, formas de preparação;
- Formato dos resultados produzidos, interpretação dos mesmos;

Sobre verificação de conformidade mais especificamente, o trabalho envolveu a pesquisa de algoritmos de análise existentes além de uma aplicação em um cenário prático. Tendo em vista os requisitos levantados, para a melhoria do procedimento de auditoria de processos de software, se conclui que eles podem ser satisfeitos (ao todo ou em parte) por técnicas automáticas de descoberta de conhecimento. A cobertura de tais requisitos é compreendida da seguinte forma:

- Auditoria não intrusiva: esse é um dos aspectos mais relevantes, visto que a análise dos dados de processos de forma automática diminui a dependência de entrevistas para obtenção de informações;
- Apresentar indicadores: durante o estudo foi possível identificar que uma das principais características de algoritmos de análise de conformidade é a apresentação de métricas e outras informações que permitam avaliar quantitativamente o grau de semelhança entre processo definido e executado. Além disso, os próprios modelos obtidos por descoberta de processos fornecem indicadores que também podem ser utilizados, juntamente com a exploração visual, na avaliação de comportamento da execução dos processos;
- Agilizar o trabalho de auditoria: considera-se que a mineração de processos também pode auxiliar nesse aspecto, já que o volume de dados analisados pode ser muito maior do que seria viável manualmente;

- Não exigir grande interferência no processo atual: pelo menos no caso da empresa parceira, algumas modificações poderiam ser necessárias, tanto no modelo quanto no registro, para obter informações mais detalhadas sobre a execução dos processos. Mesmo assim, entende-se que a rotina de trabalho dos colaboradores não seria afetada consideravelmente. Com alterações simples na forma e classificação dos registros de atividades se pode aumentar sensivelmente a rastreabilidade dos eventos ocorridos no dia-a-dia dos projetos;

Portanto, considerando os objetivos inicialmente definidos para o trabalho, as principais contribuições são relacionadas a seguir:

- discussão acerca dos problemas envolvendo análise de conformidade em PDS e requisitos de uma solução ideal;
- apresentação de métodos de mineração de processos como ferramenta para melhoria de eficiência e qualificação dos resultados de auditorias de processos de software;
- análise sob o aspecto prático da aplicação da mineração de processos, baseada em um estudo de caso exploratório com dados de uma grande empresa de software;
- compilação de uma série de sugestões para exploração de mineração de processos em PDS, produzidas a partir da experiência adquirida durante o estudo realizado;

## 6.2 Trabalhos futuros

De fato, a ordem na qual atividades são desenvolvidas em um processo fornece diversas pistas sobre o andamento desse processo e acerca da aderência de seus colaboradores com relação ao planejamento. Porém, também é fato que PDS lidam intensamente com a criação e alteração de todo o tipo de artefato (documentos, códigos, modelos, etc.). Grande parte desses processos envolve atividades desse tipo. Trata-se muitas vezes de pequenas tarefas que normalmente não são registradas explicitamente, como revisões informais ou pequenas alterações em documentos ou até mesmo em códigos. Consequentemente, se torna difícil monitorar precisamente tais atividades.

Mesmo assim é importante que tais informações também possam ser rastreadas, já que afetam o projeto diretamente. Nesse aspecto, ferramentas computacionais tipicamente utilizadas na engenharia de software podem fornecer dados importantes para uma análise ainda mais consistente sobre os processos. O conjunto de ferramentas de suporte utilizadas atualmente permite que projetos de software sejam monitorados, não só sob a visão de sequência de atividades, mas também com relação aos recursos e artefatos envolvidos. Portanto, essa é uma direção importante vista como trabalho futuro: explorar outras visões

do processo. É parte central dos modelos de desenvolvimento de software definir os papéis envolvidos em cada atividade e os produtos de entrada e saída em cada parte do processo. A mineração de processos também fornece suporte esse tipo de análise. Sistemas de gerência de configuração (SCM), por exemplo, são comumente utilizados em PDS como ferramenta para centralização de dados e controle de versões. Mas, tais ferramentas normalmente produzem logs de todos os eventos como, por exemplo, quem acessou, quais documentos, que tipo de acesso, quando foi feito. Utilizar-se do log de um sistema de SCM para complementar a análise de conformidade sob as perspectivas organizacional (recursos) e de informação (artefatos) é parte dos trabalhos futuros dessa pesquisa.

Outro ponto apontado como trabalho futuro diz respeito ao provimento de métricas mais inteligíveis sob a visão de um auditor de SQA. O presente trabalho se limitou a análise das métricas de *fitness* produzidas por algoritmos de verificação de conformidade. Contudo, se considera que a apresentação de métricas derivadas, de fácil entendimento, poderia auxiliar ainda mais a visão da equipe de SQA sobre os processos executados. Dados estatísticos como o número de instâncias inconformes, valor mínimo e máximo de *fitness*, quantidade de atividades faltantes e/ou executadas fora de ordem para cada instância seriam alguns exemplos. Na verdade todos esses indicadores podem ser obtidos indiretamente dos algoritmos. O trabalho necessário seria extrair essas informações e apresentá-las adequadamente, juntamente com aquelas já disponibilizadas.

A melhoria no procedimento de pré-processamento de dados também é uma questão a ser tratada na continuação do trabalho. Se faz necessária a implementação de uma interface com rotinas automáticas de seleção, consolidação e filtragem dos dados para facilitar essa tarefa. Tendo como objetivo principal facilitar o trabalho de auditoria em PDS, não é ideal que seja exigido um procedimento trabalhoso e complexo de preparação dos dados. Com poucos cliques, o auditor deveria ser capaz de, pelo mínimo, poder selecionar a fase e o número de instâncias a serem analisadas.

# Referências


[1] AALST, W. “The application of petri nets to workflow management”. *Journal of Circuits, Systems and Computers 8*, 1, 1998, 21–66.

[2] AALST, W. “Business alignment: using process mining as a tool for delta analysis and conformance testing”. *Requir. Eng. 10*, 3, 2005, 198–211.

[3] AALST, W.; BASTEN, T. “Identifying commonalities and differences in object life cycles using behavioral inheritance”. In *Application and Theory of Petri Nets*, vol. 2075. Berlin: Springer, 2001, pp. 35–52.

[4] AALST, W.; DONGEN, B.; HERBST, J.; MARUSTER, L.; SCHIMM, G.; WEIJTERS, A. “Workflow mining: a survey of issues and approaches”. *Data Knowl. Eng. 47*, 2, 2003, 237–267.

[5] AALST, W.; GÜNTHER, C. “Finding structure in unstructured processes: The case for process mining”. In *Seventh International Conference on Application of Concurrency to System Design*, 2007, pp. 3–12.

[6] AALST, W.; MEDEIROS, A. “Process mining and security: Detecting anomalous process executions and checking process conformance”. *Electronic Notes in Theoretical Computer Science 121*, 2005, 3–21.

[7] AALST, W.; REIJTERS, H.; WEIJTERS, A.; VAN DONGEN, B.; MEDEIROS, A.; SONG, M. e VERBEEK, H. “Business process mining: An industrial application”. *Information Systems 32*, 5, 2007, 713–732.

[8] AALST, W.; WEIJTERS, T.; MARUSTER, L. “Workflow mining: discovering process models from event logs”. *IEEE Transactions on Knowledge and Data Engineering 16*, 9, 2004, 1128–1142.

[9] AHERN, D. C.; A. TURNER, R. “Cmmi distilled: A practical introduction to integrated process improvement (3rd edition) (sei series in software engineering)”, 3 ed. Addison-Wesley Professional, May 2008.

[10] BALDONI, M.; BAROGLIO, C.; MARTELLI, A.; PATTI, V.; SCHIFANELLA, C. “Verifying the conformance of web services to global interaction protocols: A first step”. In *Lecture Notes in Computer Science*, vol. 3670. Berlin: Springer, 2005, pp. 257–271.

[11] BECKER, K.; RUIZ, D.; CUNHA, V.; NOVELLO, T.; SOUZA, F. “Spdw: A software development process performance data warehousing environment”. In *SEW ’06: Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop*, Washington, DC, USA, 2006, IEEE Computer Society, pp. 107–118.

[12] CHUNQIN, G.; CHANG, H.; YANG, Y. “Overview of workflow mining technology”. In *IEEE International Conference on Granular Computing. GRC 2007*, 2007, pp. 347–353.

[13] COALITION, W. M. “Workflow management coalition terminology & glossary (document no. wfmc-tc-1011)”. Winchester: Workflow Management Coalition, 1999.

[14] COLOMBO, A.; DAMIANI, E.; GIANINI, G. “Discovering the software process by means of stochastic workflow analysis”. *Journal of Systems Architecture 52*, 11, 2006, 684 – 692.

[15] COOK, J. WOLF, A. “Discovering models of software processes from event-based data”. *ACM Trans. Softw. Eng. Methodol. 7*, 3, 1998, 215–249.

[16] COOK, J.; HE, C.; MA, C. “Measuring behavioral correspondence to a timed concurrent model”. In *IEEE International Conference on Software Maintenance*, Los Alamitos, CA, USA, 2001, vol. 1, IEEE Computer Society, pp. 332–351.

[17] COOK, J.; WOLF, A. “Software process validation: quantitatively measuring the correspondence of a process to a model”. *ACM Trans. Softw. Eng. Methodol. 8*, 2, 1999, 147–176.

[18] CRUZ, J.; RUIZ, D. “Uma experiência em mineração de processos de manutenão de software”. In *II Workshop de Gestão de Processos de Negócio (II WBPM)*, 2008, pp. 247–253.

[19] DESEL, J. “Validation of process models by construction of process nets”. In *Business Process Management, Models, Techniques and Empirical Studies*, London, UK, 2000, Springer-Verlag, pp. 110–128.

[20] DUTSDAR, S.; GOMBOTZ, R. “Discovering web service workflows using web services interaction mining”. *International Journal on Business Process Integration and Management 1*, 4, 2006, 256–266.

[21] ELLIS, W. “Dynamic change within workflow systems”. In *Proceedings of the Conference on Organizational Computing Systems*, 1995.

[22] EMAN, K.; MELO, W.; DROUIN, J. “Spice: The theory and practice of software process improvement and capability determination”, vol. 1. Los Alamitos, CA: IEEE Computer Society Press, 1997.

[23] FAGAN, M. “Advances in sofware inspections”. *IEEE Transactions on Software Engineering 12(7)*, 1986, 744–751.

[24] FERREIRA, A.; SANTOS, G.; CERQUEIRA, R.; MONTONI, M.; ROCHA, A. “Applying iso 9001: 2000, mps.br and cmmi to achieve software process maturity: Bl informatica’s pathway”. In *ICSE ’07: Proceedings of the 29th international conference on Software Engineering*, Washington, DC, USA, 2007, IEEE Computer Society, pp. 642–651.

[25] GLABBEEK, R.; WEIJLAND, W. “Branching time and abstraction in bisimulation semantics”. *Journal of the ACM 43*, 3, 1996, 555–600.

[26] GROUP, P. M. “Process mining research tools application”. Disponível em: http://www.processmining.org, Acesso em: novembro 2009.

[27] GÜNTHER, C.; AALST, W. “Fuzzy mining: Adaptive process simplification based on multi-perspective metrics”. In *International Conference on Business Process Management (BPM 2007)*, 2007, vol. 4714, pp. 328–343.

[28] HAN, J.; KAMBER, M.; PEI, J. “Data mining: Concepts and techniques, second edition (the morgan kaufmann series in data management systems)”, 2 ed. San Francisco:Morgan Kaufmann, 2006.

[29] HARDGRAVE, B.; ARMSTRONG, D. “Software process improvement: it’s a journey, not a destination”. *Communications of ACM 48*, 11, 2005, 93–96.

[30] HERBST, J. “Inducing workflow models from workflow instances”. In *Concurrent Engineering Europe Conference*, 1999, pp. 175–182.

[31] HUMPHREY, W. “A discipline for software engineering”. Addison-Wesley, 1989.

[32] HUO, M.; ZHANG, H.; JEFFERY, R. “An exploratory study of process enactment as input to software process improvement”. In *WoSQ ’06: Proceedings of the 2006 international workshop on Software quality*, 2006, pp. 39–44.

[33] IBM. “Rational unified process: Best practices for software development teams”. disponível em: http://www.ibm.com/developerworks/rational/library/253.html., Acesso em: novembro 2009.

[34] INSTITUTE, S. S. E. “Cmmi for development, version 1.2”. Pittsburgh: Carnegie Mellon University and Software Engineering Institute, 2006.

[35] KAN, S. “Metrics and models in software quality engineering (2nd edition)”. Addison-Wesley Professional, September 2002.

[36] KURNIAWATI, F.; JEFFERY, R. “The use and effects of an electronic process guide and experience repository: a longitudinal study”. *Information and Software Technology 48*, 7, July 2006, 566–577.

[37] M. CHRISSIS, M. K. e. S. S. “Cmmi : Guidelines for process integration and product improvement”, 1 ed. Boston:Addison-Wesley, 2003.

[38] MEDEIROS, A.; AALST, W.; WEIJTERS, A. “Workflow mining: Current status and future directions”. In *On The Move to Meaningful Internet Systems*, vol. 2888. Berlin:Springer, 2003, pp. 389–406.

[39] MEDEIROS, A.; WEIJTERS, A. “Process equivalence: Comparing two process models based on observed behavior”. In *International Conference on Business Process Management (BPM 2006)*, 2006, vol. 4102 of *Lecture Notes in Computer Science*, pp. 129–144.

[40] MURAFELIJA, B. STROMBERG, H. “Process improvement with cmmi v1.2 and iso standards”, 1 ed. Boca Raton:Auerbach Publications, 2008.

[41] PRESSMAN, R. “Software engineering”. New York: McGraw-Hill, 2004.

[42] ROZINAT, A.; AALST, W. “Conformance checking of processes based on monitoring real behavior”. *Information Systems 33*, 1, 2008, 64–95.

[43] RUBIN, V.; GÜINTER, C.; AALST, W.; KINDLER, E.; DONGEN, B.; SCHÄFER, W. “Process mining framework for software processes”. In *Software Process Dynamics and Agility*. Berlin:Springer, 2007, pp. 169–181.

[44] RUSSEL, N.; TER HOFSTEDE, A.; MULLYAR, N. “Workflow control flow patterns: A revised view. [s.1], 2006. 134 p.”. http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf, Acesso em: novembro 2009.

[45] SILVA, L.; TRAVASSOS, G. “Tool-supported unobtrusive evaluation of software engineering process conformance”. In *International Symposium on Empirical Software Engineering, 2004. ISESE ’04.*, 2004, pp. 127–135.

[46] SORUMGARD, S. *Verification of Process Conformance in Empirical Studies of Software Development*. PhD thesis, Norwegian University of Science and Technology, 1997.

[47] VAN DONGEN, B.; AALST, W. “A meta model for process mining data”. In *Conference on Advanced Information Systems Engineering*, Porto, Portugal, 2005, vol. 161.

[48] VAN DONGEN, B.; MEDEIROS, A.; VERBEEK, H.; WEIJTERS, A.; AALST, W. “The prom framework: A new era in process mining tool support”. In *26th International Conference, ICATPN 2005*, June 2005, vol. 3536, Springer Verlag, pp. 444–454.

[49] WEBER, K.; ARAUJO, E.; ROCHA, A.; MACHADO, C.; SCALET, D.; SALVIANO, C. “Brazilian software process reference model and assessment method”. *Computer and Information Sciences - ISCIS 2005 3733*, 2005, 402–411.

[50] WEIJTERS, A.; AALST, W. “Rediscovering workflow models from event-based data using little thumb”. *Integrated Computer-Aided Engineering 10*, 2003, 151–162.

[51] WEN, L.; WANG, J.; SUN, J. “Detecting implicit dependencies between tasks from event logs”. In *APWeb 2006, LNCS 3841*. Berlin: Springer-Verlag, 2006, pp. 591–603.

[52] WHITEHEAD, J. “Collaboration in software engineering: A roadmap”. In *FOSE ’07: 2007 Future of Software Engineering*, Washington, DC, USA, 2007, IEEE Computer Society, pp. 214–225.

# A. Apêndice - Modelos de processos

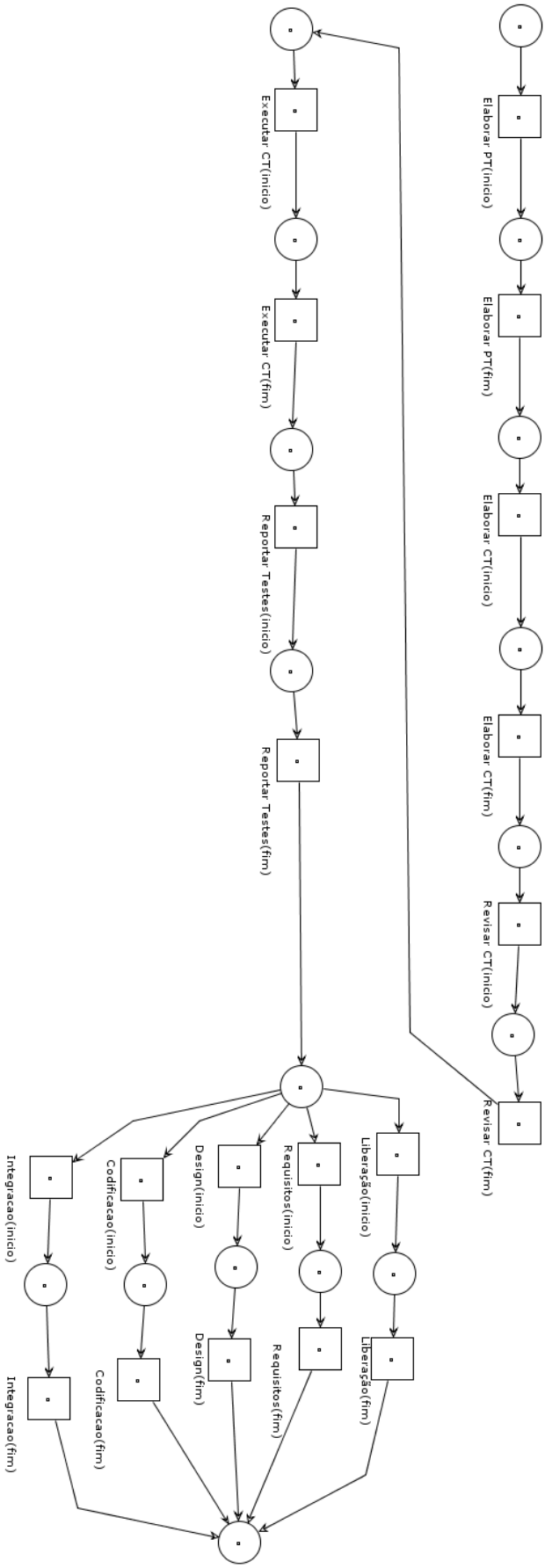


Figura A.1: Modelo do processo de testes desenhado em rede de Petri

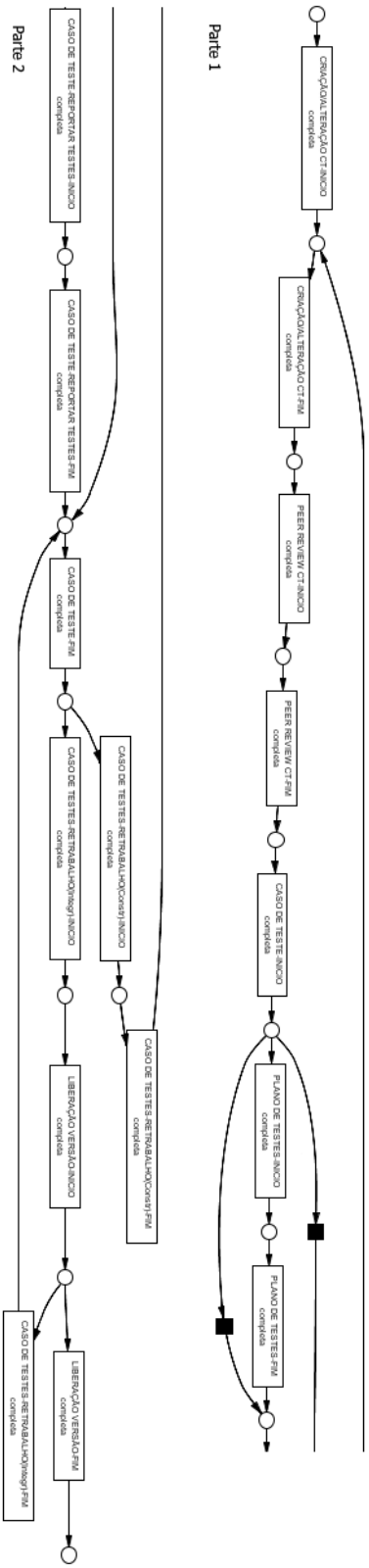


Figura A.2: Modelo obtido por mineração com o algoritmo Heuristics Miner transformado para rede de Petri