UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

JEAN MICHEL LAU

Descoberta e Análise de Associações entre Padrões de Atividade em Modelos de Processos de Negócio

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Cirano Iochpe
Orientador

Porto Alegre, junho de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Lau, Jean Michel

Descoberta e Análise de Associações entre Padrões de Atividade em Modelos de Processos de Negócio / Jean Michel Lau – Porto Alegre: Programa de Pós-Graduação em Computação, 2009.

88 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: Cirano Iochpe.

1.Processos de negócio. 2.*Workflow* 3.Padrões. 4.Modelagem. 5.Reuso. I. Iochpe, Cirano. II. Título.

AGRADECIMENTOS

Agradeço a minha família pelo apoio e suporte para que eu pudesse atingir esta conquista.

Agradeço ao meu orientador e aos integrantes do grupo de pesquisa por todas as ideias, sugestões, críticas e comentários que tornaram o desenvolvimento deste trabalho interessante e desafiador.

Agradeço à UFRGS e ao Instituto de Informática pelo ensino gratuito e de qualidade.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BPEL4WS	Business Process Execution Language for Web Services
BPMI	Business Process Management Initiative
BPMN	Business Process Modeling Notation
CASE	Computer Aided Software Engineering
DCBD	Descoberta de Conhecimento em Bases de Dados
FSG	Frequent Sub-Graph
KDD	Knowledge Discovery in Databases
OMG	Object Management Group
PAIS	Process Aware Information System
SGBD	Sistema Gerenciador de Banco de Dados
SGW	Sistema Gerenciador de Workflow
TI	Tecnologia da Informação
WfMC	Workflow Management Coalition
WfMS	Workflow Management System

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A tecnologia de *workflow* tem-se mostrado cada vez mais importante para o mercado de automação de processos e para que as organizações desempenhem suas funções da melhor maneira possível (MUTSCHLER, 2008). A modelagem de processos ganhou em importância neste contexto, tornando-se alvo de pesquisas, inclusive acadêmicas. Nestas pesquisas, certas estruturas recorrentes que precisam ser redesenhadas a cada nova modelagem foram identificadas. Conhecidas também como padrões, existem com variados focos e objetivos (AALST, 2003), (RUSSEL, 2004a), (RUSSEL, 2004b), (BRADSHAW, 2005), (RUSSEL, 2006), (THOM, 2006a), (THOM, 2009). O conjunto de padrões apresentados em Thom (2006a), (2009) é interessante, pois apresenta padrões de atividade os quais representam funções de negócio recorrentes, tais como aprovação de documentos, notificação de pagamento e solicitação de execução de tarefa. Estes padrões são mais próximos do nível de descrição dos especialistas do domínio e, por isto, mostram-se como bastante promissores para serem aplicados durante a análise e modelagem de processos.

Diversos pesquisadores defendem a ideia de que a padronização de estruturas de processo e a re-utilização dos padrões resultantes deste exercício podem incrementar a produtividade, tanto na fase de modelagem e documentação dos processos como na fase de manutenção ou de re-escrita destes mesmos. A utilização de padrões de *workflow* pode tornar a fase de modelagem mais eficiente e de melhor qualidade, gerando processos mais bem formatados e menos suscetíveis a erros (OBJECT MANAGEMENT GROUP, 2006).

Nesta direção, este trabalho pretende utilizar os padrões de atividade desenvolvidos em Thom (2006a) para descobrir e analisar associações recorrentes entre eles. Com estas informações pode-se melhorar a fase de modelagem de processos na medida em que será possível saber como os padrões de atividade costumam conectarem-se uns com os outros. Estas prováveis associações podem então ser sugeridas, de forma automática, a um usuário no momento da modelagem de um novo processo. Neste trabalho serão buscados meios que possibilitem obter estas informações, contemplando a adoção de uma metodologia, algoritmos e a adaptação destes ao problema proposto.

Ao final, espera-se obter subsídios para aprimorar e difundir a modelagem com base no reuso. Serão buscadas informações que permitam avaliar se a fase de modelagem pode se beneficiar da utilização de padrões e proporcionar resultados interessantes. A partir do conhecimento das associações recorrentes entre os padrões de atividade, estes poderão ter a sua utilização aumentada e a modelagem de processos com a utilização de padrões será evoluída. Desta forma, a utilização inteligente de estruturas recorrentes na modelagem de processos estará sendo ampliada e promovida.

Palavras-Chave: Processos de negócio, w*orkflow*, padrões, modelagem, reuso.

Discovery and Analysis of Associations between Activity Patterns in Business Process Models

ABSTRACT

Workflow technology has been increasingly important for the process automation market so that organizations can perform their duties in the best possible way (MUTSCHLER, 2008). Process modeling has gained in importance in this context, becoming subject of research, including academic. In these studies, certain recurrent structures that must be redesigned for each new modeling were identified. Also known as patterns, they exist with different focuses and objectives (AALST, 2003), (RUSSEL, 2004a), (RUSSEL, 2004b), (BRADSHAW, 2005), (RUSSEL, 2006), (THOM, 2006a), (THOM, 2009). The patterns set presented in Thom (2006a), (2009) is interesting, because it shows activity patterns which represent recurrent business functions, such as document approval, payment notification and task execution request. These patterns are closer to the domain experts description level and, therefore, are as much promising to be used during process analysis and modeling.

Several researchers advocate the idea that the standardization of process structures and the reuse of the resulting patterns from this exercise can increase the productivity, both at the process modeling and documentation phases as in the maintenance or re-write phases of these. The use of workflow patterns can make modeling phase more efficient and with better quality, generating more well-formed and less error prone processes (OBJECT MANAGEMENT GROUP, 2006).

In this direction, this work intends to use the activity patterns developed in Thom (2006a) to find and analyze recurrent associations between them. With this information it is possible to improve process modeling phase as it will be possible to know how activity patterns are usually connected with each other. These probable associations can then be suggested, in an automatic way, to a user when modeling a new process. In this research, mechanisms to enable to obtain this information, including the adoption of a methodology, algorithm and the adaptation of these to the proposed problem will be seek.

At the end, it is expected to obtain subsidies to improve and disseminate the modeling based on reuse. Information will be sought to assess whether the modeling phase can benefit from pattern utilization and provide interesting results. From the recurrent association between activity patterns knowledge, they may have its utilization increased and the process modeling with the use of patterns will be evolved. Thus, the intelligent use of recurrent structures will be expanded and promoted.

Keywords: Business processes, workflow, patterns, modeling, reuse.

1 INTRODUÇÃO

A utilização de *software* para o controle e gerenciamento dos processos de negócio das organizações tem sido fator primordial para que estas atinjam competitividade, sejam flexíveis e solícitas a seus clientes e consigam realizar um melhor controle sobre suas operações. Recentemente, estas companhias têm demonstrado um crescente interesse no melhor gerenciamento, organização, controle e otimização de seus processos de negócio (MUTSCHLER, 2008).

As organizações modernas, de forma geral, buscam realizar seus propósitos de negócio da melhor maneira e no menor tempo possível, a fim de satisfazer a seus clientes e abrir a possibilidade de atender a um maior número deles (CASATI, 2004). Como é de se esperar, as organizações são criadas para atingir certas metas e objetivos, os quais têm um impacto direto na estrutura e comportamento destas. Em virtude disto, diferentes partes acabam sendo, cada uma, especializadas em cumprir determinadas metas individuais. Um exemplo é a tarefa da equipe de recursos humanos, a qual se relaciona com os objetivos do sistema da seguinte maneira: eles são responsáveis por prover recursos qualificados para realizar as tarefas dentro de uma empresa, mas não são responsáveis por objetivos de produção, os quais cabem a outros setores da organização. Com um crescente número de participantes especializados em diferentes etapas dos processos, a coordenação destes como um todo se torna extremamente importante (THOM, 2002).

Neste contexto, sistemas PAIS (*Process-Aware Information Systems*) apresentam-se como ferramentas alternativas para o desenho e a automação de processos de negócio (MUTSCHLER, 2008). A utilização da tecnologia de *workflow* para o gerenciamento destes processos tem-se tornado cada vez mais popular e importante para que as organizações atinjam seus objetivos de negócio. A exigência de respostas rápidas faz com que os processos atualmente executados sejam modelados para, então, serem gerenciados através de um sistema gerenciador de *workflow* (SGW) (INTALIO, 2006), (OBJECT MANAGEMENT GROUP, 2006), (THOM, 2006a).

Nos últimos anos, para se manterem competitivas, muitas organizações têm explorado técnicas da abordagem de gestão por processos. Tal abordagem recebeu impulso adicional através da norma ISO 9001:2000, a qual define que a organização deve ser retratada por seus processos de negócio principais e não pelo seu organograma (FUNDAÇÃO NACIONAL DA QUALIDADE, 2006).

A gestão por processos associada à tecnologia de workflow pode trazer diversos benefícios à organização, tais como: (a) descrição precisa e não ambígua dos processos de negócio existentes; (b) melhoria na definição de novos processos; (c) maior eficácia na coordenação do trabalho entre diferentes participantes dos processos; (d) obtenção, em tempo real, de informações precisas sobre o andamento dos processos; e (e)

padronização dos processos executados, de forma manual ou automatizada, pela organização (THOM, 2007a).

A tecnologia de *workflow*, através da automação dos processos de negócio das organizações, contribui para a redução de custos de projeto, implementação e documentação de processos. Contribui, também, para a redução no tempo de execução dos mesmos. Além disso, erros e redundâncias na execução dos processos podem ser identificados e removidos ou evitados quando da utilização de estruturas padrão. Ao mesmo tempo, ela aumenta o controle sobre os processos da organização levando ao incremento da qualidade e produtividade dos processos, dos seus resultados e da organização como um todo (IOCHPE, 2001), (MUEHLEN, 2004), (THOM, 2007b).

1.1 Motivação

Uma das principais fases do desenvolvimento de um sistema de *workflow* é a etapa de modelagem dos processos de negócio da organização (OBJECT MANAGEMENT GROUP, 2006). Esta etapa é tanto mais importante na medida em que as organizações, cada vez mais, expandem-se e interagem entre si, com seus fornecedores, clientes e parceiros. A partir do esforço de modelagem é que os processos são documentados formalmente e suas interfaces com outros processos e serviços são identificadas e objetivamente projetadas.

A modelagem ou o desenho do processo fornece um meio simples de representar o que é executado ao longo do mesmo de forma que seja fácil a sua compreensão e sua, eventual, melhoria (otimização). Isto beneficia tanto os analistas que criam os modelos iniciais dos processos, quanto os desenvolvedores técnicos responsáveis pela implementação da tecnologia que irá executá-los, além das pessoas que irão gerenciar e monitorar a execução destes processos (OBJECT MANAGEMENT GROUP, 2006).

Um esforço na padronização da modelagem de processos tem sido realizado pela *Business Process Management Initiative* (BPMI) através da *Business Process Modeling Notation* (BPMN), a qual propõe notações para serem utilizadas na fase de modelagem, e também provê um mapeamento formal para uma linguagem de execução de processos de negócio como a BPEL4WS (*Business Process Execution Language for Web Services*). Deste modo, tanto a funcionalidade como a infra-estrutura da organização podem ser captadas e representadas, gráfica e textualmente, através da BPMN, possibilitando uma comunicação padronizada de seus procedimentos e maior eficiência na sua realização, garantindo, inclusive, meios para que os processos modelados sejam adaptados mais facilmente em casos de modificações estruturais na organização (OBJECT MANAGEMENT GROUP, 2006).

No contexto da modelagem de processos, seja com BPMN ou outras notações, existe um esforço de pesquisa para identificar estruturas recorrentes que possam ser consideradas padrões de modelagem (ex.: solicitação de execução de tarefa, notificação de um resultado, tomada de decisão). Pesquisadores esperam que a identificação e a re-utilização de padrões de modelagem, em *workflow*, possam ser aplicadas para melhorar a qualidade e a produtividade na fase de desenvolvimento dos modelos dos processos (OASIS, 2006). Em vista disto, várias pesquisas neste campo de estruturas reutilizáveis têm sido desenvolvidas buscando identificar e difundir construções que aparecem frequentemente nos processos modelados (AALST, 2003), (RUSSEL, 2004a), (RUSSEL, 2004b), (BRADSHAW, 2005), (RUSSEL, 2006), (THOM, 2006a), (THOM, 2009). Contudo, até este momento, a maioria, senão a totalidade destas pesquisas, não

aborda as combinações destes padrões nem a frequência com que estas se fazem presentes em modelos de processos da mesma ou de diferentes organizações. Por outro lado, observa-se que o suporte a padrões ainda não é efetivamente explorado pelas ferramentas de modelagem atualmente existentes (KIEPUSZEWSKI, 2003), (CUNTZ, 2004), (INTALIO, 2006), sendo que, em muitas vezes, este suporte é praticamente inexistente. Não há funcionalidades que permitam aos analistas definir, consultar e reutilizar padrões de *workflow* adequadamente, sobretudo aqueles relacionados a funções recorrentes em processos de negócio. Também não é comum uma ferramenta capaz de oferecer apoio ao desenhista de processos, oferecendo a ele sugestões de "próximos" padrões de modelagem a serem inseridos em um modelo parcialmente construído até um certo momento.

Em Thom (2006b; e, 2009) é apresentado um conjunto de padrões que representam tipos recorrentes de atividades em processos de *workflow*. Estes "padrões de atividade" são representados como padrões de atividade de bloco, conforme estes são previstos pela WfMC (WORKFLOW MANAGEMENT COALITION, 1999a). Os padrões de atividade de bloco da WfMC, os quais são referidos, daqui para a frente, simplesmente como padrões de atividade, podem representar a execução atômica de atividades complexas.

Cada padrão proposto em Thom (2006b) refere-se a uma função de negócio recorrente, frequentemente encontrada em processos de negócio. A mineração e a análise, feitas manualmente, de um conjunto de 190 modelos de processos de *workflow* evidenciou a existência deste conjunto de padrões com alto *Suporte* além de ter revelado que o conjunto de padrões, juntamente com os padrões de controle de fluxo, é necessário e suficiente para modelar todos os 190 processos que foram investigados neste estudo de caso. Além disso, a pesquisa de Thom (2006b) também resultou em um conjunto de regras associativas. Estas regras não somente ajudam a melhor definir padrões de atividades específicos de *workflow*, como também os combinam com os padrões existentes de controle de fluxo (AALST, 2003) e podem ser úteis na construção de modelos mais complexos.

A semântica destes padrões e a forma como podem ser representados (UML, Cálculo Pi) permitem que os mesmos sejam usados como "macro-atividades" que realizam funções mais complexas, tal como a aprovação de um documento, por exemplo. Estas macro-atividades evidenciam padrões de atividades complexas que necessitam ser atomicamente executadas dentro de um processo de *workflow*, diferindo dos padrões que representam apenas controle de fluxo (baseados em eventos ou dados) propostos em Aalst (2003) e Russel (2004a). Apesar destas atividades mais complexas serem recorrentes em processos de negócio de muitas organizações, elas necessitam ser reiteradamente redesenhadas, pois não existe nem conhecimento teórico, nem estudos de caso suficientes para a construção de editores de processo com funcionalidades de suporte à modelagem baseadas na reutilização de padrões de *workflow*.

Além disto, por serem definidos em um nível de abstração mais alto do que, por exemplo, padrões de roteamento, os padrões de atividade propostos em Thom (2006b) estão mais próximos das descrições de atividades reais como estas são feitas pelos analistas de negócio ou profissionais (não da área da tecnologia de informação) da organização. Desta maneira, estudar relacionamentos entre estes padrões é interessante, pois o resultado deste estudo poderia ser mais facilmente aplicado por desenhistas de processos de negócio.

Ao trabalhar desta maneira, ao descrever um processo que possui uma etapa de aprovação de documentos, um analista ou projetista não necessitará referir-se ao fato de que existe um documento que passa entre diversas pessoas, as quais devem lê-lo e, após, dar seu parecer. O analista apenas dirá que, neste ponto do processo, ocorre uma "aprovação" de um documento. Esta abstração é algo que os padrões de atividade conseguem atingir. Além de trazer uma grande agilidade para o desenho (aumento da produtividade), os padrões auxiliam na documentação dos processos e em suas leitura e compreensão por profissionais que não são o autor da modelagem.

Com base no que foi exposto, espera-se contribuir com a pesquisa de uma nova forma de modelagem de processos. Com esta nova forma, vislumbra-se o ponto em que o usuário seja auxiliado e possa usufruir dos padrões da melhor maneira possível. Isto pode tanto dar-se através da extensão de alguma ferramenta de modelagem já existente, como a ADEPT (REICHERT, 2003), EPC Tools (MENDLING, 2005) ou Intalio (INTALIO, 2006), ou na forma de uma nova ferramenta de desenho de processos que contemple os objetivos desejados e aplique os resultados obtidos nesta pesquisa.

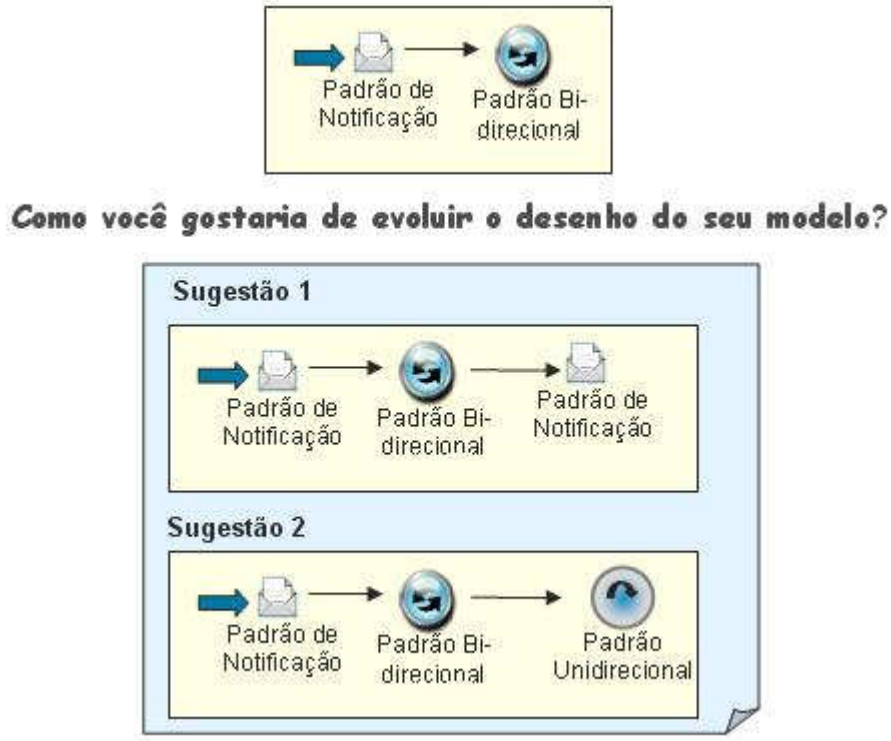

Figura 1.1: Sugestão automática de evolução de modelagem

Neste contexto, apresenta-se a oportunidade para a pesquisa e o desenvolvimento de soluções envolvendo os padrões de atividade e sua respectiva disponibilização para a fase de modelagem. Buscando aprimorar ainda mais esta etapa e tirar maior proveito do uso dos padrões, a "sugestão automática" de padrões, na medida em que um processo está sendo modelado, é uma funcionalidade bastante interessante a ser pesquisada. Esta sugestão automática refere-se à possibilidade de inferir certo conhecimento sobre o processo sendo modelado e a capacidade de um sistema de computador sugerir como este modelo deve ou pode ter o seu desenho evoluído, conforme ilustrado na Figura 1.1. Com a utilização dos padrões de atividade como base, o sistema de computador poderia sugerir qual o próximo padrão de atividade, ou sequência de padrões, a serem inseridos em um modelo e guiar o usuário no desenho deste. Em vista disto, a motivação desta

dissertação de mestrado é a pesquisa de estratégias para agregar funcionalidade às ferramentas de modelagem a fim de que possam auxiliar o usuário na reutilização de estruturas de *workflow* recorrentes em processos de negócio, como as estruturas propostas em Thom (2006b).

A ideia de uma ferramenta ao estilo das ferramentas CASE (*Computer Aided Software Engineering*) da engenharia de software para a modelagem de *workflow* faz imaginar como esta etapa poderia ser beneficiada caso existisse uma ferramenta capaz de auxiliar um usuário no momento em que este está desenhado um processo de negócio. Surgem questões procurando investigar se é possível desenvolver algo neste sentido, como isto pode ser feito, quais os benefícios que podem ser atingidos, como medir se os benefícios foram alcançados, entre outras. Para que tal ferramenta seja possível de existir, é necessário desenvolver estratégias para identificar quais os relacionamentos mais frequentes entre os diferentes tipos de padrões de atividade. Este conhecimento, o qual provavelmente possa ser detalhado com base em diferentes categorias de processo, deve se constituir na base para a funcionalidade de predição de uma ferramenta de modelagem de *workflow*.

1.2 Objetivos

A pesquisa, adaptação e aplicação de uma estratégia adequada à descoberta e análise das informações necessárias à disponibilização e sugestão automática de padrões de atividade durante o processo de modelagem é o grande objetivo deste trabalho. Para que este objetivo maior seja atingido, a alternativa a ser aplicada é utilizar-se de técnicas de inteligência artificial para a descoberta de conhecimento em bases de dados, a fim de identificar a maneira como os padrões de atividade estão associados uns com os outros em uma série de modelos. Esta escolha dá-se pelo desejo de seguir um processo estruturado e amplamente conhecido em detrimento de uma busca empírica e manual por recorrências de associações entre padrões de atividade.

No futuro, de posse desta informação, ferramentas de modelagem poderão consultar este conhecimento e orientar o processo de modelagem, oferecendo, ao usuário, os mais prováveis próximos padrões, ou sequência de padrões, que ele necessita inserir em seu modelo. Ou seja: se o modelo do usuário já possui os padrões A, B e C conectados de uma certa maneira, a ferramenta poderá, a partir de uma consulta a uma base de conhecimento, sugerir as alternativas de próximas estruturas de *workflow* para o modelo do usuário, o qual poderá optar por uma delas e incluí-la no desenho, detalhando-a em um segundo momento com informações específicas do seu negócio. Em virtude deste objetivo, técnicas de descoberta de conhecimento serão estudadas e avaliadas para que a mais adequada seja selecionada e aplicada para a obtenção dos resultados desejados.

Em trabalhos anteriores (THOM, 2006b) já foram mineradas regras associativas para a descoberta de como a existência de um padrão em um modelo de processo pode condicionar a existência de um padrão previamente conhecido em sua sequência. Contudo, este estudo foi desenvolvido apenas para um padrão como antecedente e um outro como consequente imediato.

No presente trabalho este conhecimento será aprimorado, obtendo informações de associações entre padrões em níveis mais detalhados. Ou seja, considerando-se um modelo com um conjunto de padrões conectados de certa maneira como antecedente, deve-se poder indicar, com diferentes graus de certeza, qual o padrão e onde este deve ser inserido para dar continuidade a sua modelagem.

A estratégia de descoberta de conhecimento a ser selecionada deve permitir identificar como os padrões de atividade apresentados em Thom (2006b; e, 2009) estão relacionados uns com os outros em conjuntos quaisquer de processos de negócio. Dado um conjunto de padrões *A*, *B*, *C* e *D*, interessa identificar relacionamentos que indiquem que um processo composto pelo padrão *A* conectado com o padrão *B* possui, também, um padrão *C* conectado em uma determinada quantidade de casos analisados, conforme ilustrado no exemplo da Figura 1.2.

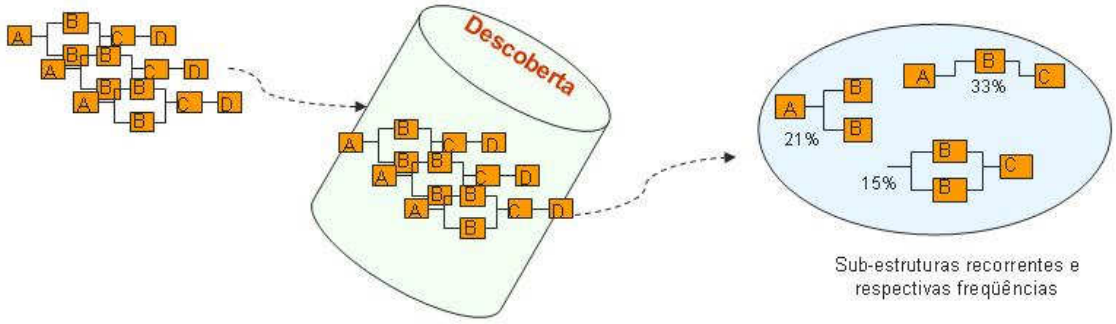

Figura 1.2: Descoberta de relacionamentos entre os padrões de atividade

A estratégia a ser selecionada deve também permitir "fatorar" as predições de próximos padrões mais prováveis por diferentes categorias de processos (ex.: processos de execução predominantemente manual versus processos mais automatizados). Com isto, poderá ser identificado se as recorrências entre os padrões de atividade mais frequentes variam entre diferentes grupos de processos e se determinados grupos possuem um conjunto de recorrências mais característico. Esta constatação é importante, pois, se confirmada, possibilita que este tipo de informação também seja utilizado futuramente em ferramentas que auxiliem automaticamente o usuário na modelagem de processos.

No esforço de adaptação de uma técnica de DCBD para a identificação e análise de co-ocorrências entre padrões de *workflow*, fez-se também necessário o desenvolvimento de uma aplicação de pós-processamento dos resultados da mineração de dados. Esta aplicação filtra resultados da mineração e, com base em informações previamente descobertas, monta listas de sugestões de evoluções mais prováveis para um modelo parcialmente desenhado. Ela automatiza algumas tarefas de análise de como a sugestão automatizada de padrões pode afetar o processo de modelagem e complementa a suíte de ferramentas que dão suporte à estratégia selecionada sendo considerada uma das principais contribuições deste trabalho, a qual é descrita, em detalhes, nesta dissertação.

1.3 Estrutura do Trabalho

O texto que segue está estruturado em sete capítulos, incluindo esta introdução. O Capítulo 2 apresenta uma breve introdução a alguns conceitos básicos de processos de negócio e *workflow*, bem como apresenta a noção de padrões e os padrões de atividade conforme Thom (2006b). Neste Capítulo também são apresentados alguns trabalhos relacionados ao suporte de padrões na modelagem.

O Capítulo 3 introduz conceitos da inteligência artificial relativos ao processo de descoberta de conhecimento. São apresentadas as suas etapas e as principais técnicas conhecidas conforme os objetivos desejados. É dada ênfase especial à etapa de

mineração de dados, por tratar-se esta de uma das mais importantes e estar fortemente relacionada com os resultados esperados.

Em vista da importância da etapa de mineração de dados, a seleção da técnica e do algoritmo a serem utilizados é abordada no Capítulo 4. São investigadas as principais características do problema em questão e é decidido como este será resolvido. Então a técnica de mineração a ser utilizada é abordada em mais detalhes, quando o algoritmo escolhido também é apresentado e comentado.

Tendo-se definido como resolver o problema, o Capítulo 5 apresenta o processo executado para efetivamente resolvê-lo. Neste capítulo as etapas do processo de descoberta de conhecimento são descritas até a mineração de dados. O conjunto de modelos de processo utilizados é apresentado e é minerado para que o conhecimento desejado seja extraído.

No Capítulo 6 tem-se a apresentação e análise dos resultados obtidos na resolução do problema proposto. Inicialmente os resultados da mineração de dados são manualmente analisados. Em seguida, evidencia-se a necessidade do desenvolvimento de uma ferramenta que automatize parte do processo e a arquitetura desta ferramenta é então apresentada. Com a ferramenta, diferentes análises são possibilitadas. Estas são apresentadas em sequência. Depois, é realizado mais um estudo com a aplicação de determinados filtros sobre a informação descoberta para verificar se a utilização desta pode ser melhorada. Então, apresenta-se um estudo de caso realizado para verificar como as associações recorrentes entre os padrões de atividade podem ser afetadas quando os processos são segmentados em diferentes categorias.

Por fim, o Capítulo 7 apresenta algumas conclusões, publicações e sugestões de trabalhos futuros. Neste capítulo as conquistas e os resultados da presente pesquisa são resumidos e ressaltados e possibilidades de expansão e melhorias são apontadas.

2 PROCESSOS DE NEGÓCIO, WORKFLOW E PADRÕES DE ATIVIDADE

A utilização de *software* para o controle e gerenciamento dos processos de negócio de uma organização tem sido fator primordial para que esta atinja competitividade, seja flexível e solícita a seus clientes e consiga realizar um melhor controle sobre suas operações. Recentemente, as companhias têm mostrado cada vez maior interesse no melhor gerenciamento, organização, controle e otimização de seus processos (MUTSCHLER, 2008).

A globalização dos mercados mundiais, sobretudo com os avanços da tecnologia em geral e da Internet, requer esforços contínuos na adaptação das novas lógicas de negócios. Qualquer um que ignore este fato encontrará dificuldades para atingir o sucesso neste ambiente de tamanha competitividade (THOM, 2006c).

Dentro deste contexto, a tecnologia de *workflow* tem-se mostrado bastante eficiente para que os objetivos organizacionais sejam atingidos, dirigindo e auxiliando a automação dos processos de negócio e facilitando a sua execução e controle. Em vista disto, este Capítulo dará uma breve introdução ao conceito de processos de negócio, à tecnologia de *workflow* e, subsequentemente, aos padrões de atividade que serão alvo de estudo do presente trabalho.

2.1 Processos de Negócio e Tecnologia de *Workflow*

Segundo a WfMC (*Workflow Management Coalition*)[1] (WORKFLOW MANAGEMENT COALITION, 1999a), um **processo de negócio** é um conjunto de um ou mais procedimentos relacionados, os quais coletivamente contribuem para a realização de um objetivo de negócio, normalmente dentro de uma estrutura organizacional definindo papéis e relacionamentos. Ainda conforme esta definição, uma entidade como esta deve ter previamente especificadas suas condições e eventos necessários para inicialização, assim como as saídas que são esperadas quando do seu

[1] A WfMC, fundada em agosto de 1993, é uma organização internacional sem fins lucrativos de produtores de *workflow*, usuários, analistas e grupos universitários e de pesquisa. Sua missão é promover e desenvolver o uso de *workflow* através do estabelecimento de padrões para terminologia de *software*, interoperabilidade e conectividade entre produtos de *workflow* (WORKFLOW MANAGEMENT COALITION, 1999b).

término. Em outras palavras, um processo de negócio é uma ordem parcial de atividades onde cada uma delas contribui em um estágio do processo para que uma organização atinja os seus propósitos (THOM, 2006b). Dentre os propósitos de negócio os quais uma organização busca, têm-se alguns como renda, retorno de investimento (*Return On Investiment* – ROI), ganho de comissões, crescimento, estabilidade, qualidade, maior utilização de capacidade produtiva, menos itens em estoque, baixo tempo de inatividade e curtos períodos de armazenamento, por exemplo (AIBER, 2004) (MUEHLEN, 2004).

Até os anos 70, a separação funcional de tarefas dentro das organizações era apropriada para as condições de mercado existentes. A partir de então, um aumento na segmentação de mercado e ciclos de vida de produção menores, entre outros fatores, fizeram com que empresários e pesquisadores buscassem estruturas organizacionais mais adequadas às condições de mercado e infra-estrutura das empresas. Em vista disto, os processos de negócio tornaram-se um ponto bastante focado na pesquisa organizacional dando origem às organizações orientadas a processos (MUEHLEN, 2004), (THOM, 2002). Tal abordagem ainda recebeu impulso adicional através da norma ISO 9001:2000, a qual define que a organização deve ser retratada por seus processos de negócio principais e não pelo seu organograma (FUNDAÇÃO NACIONAL DA QUALIDADE, 2006).

Para que se aproveite da melhor maneira possível os benefícios de uma organização orientada a processos, manutenção e controle contínuos sobre os processos de negócio são necessários. O gerenciamento de processos é a área que trata da efetiva e eficiente execução destes. Ela consiste das fases de planejamento, implementação, execução e controle dos processos, definindo um ciclo de vida que proporciona um aprimoramento contínuo. Além disso, atende aos requisitos das companhias para estarem adaptáveis às mudanças ambientais e internas. Simultaneamente, ajuda as companhias a atingir ganhos através da exploração de caminhos com eficientes custo-benefício para produzir bens e realizar serviços (MUEHLEN, 2004).

Como é de se esperar, o gerenciamento de organizações orientadas a processos requer métricas apropriadas para verificar e validar a produtividade de um processo da organização. O controle de processos deve se esforçar para garantir a racionalidade do tomador de decisão fornecendo-lhe informações relevantes sobre a execução do processo. Uma das tarefas centrais para o sucesso do controle de processos é a instalação e manutenção de uma infra-estrutura que garanta excelência operacional provendo esta informação necessária (AVERSON, 1998).

Para manterem-se competitivas, as organizações modernas estabelecem demandas de desempenho relacionadas com o tempo de execução e o consumo de recursos dos seus processos de negócio e a tecnologia de ***workflow*** tem se demonstrado como uma facilitadora auxiliando estes procedimentos. De acordo com a WfMC (WORKFLOW MANAGEMENT COALITION, 1999a), o termo *workflow* é definido como a automação de parte ou de todo um processo de negócio, no qual, documentos, informações e/ou atividades são passados de um participante a outro, a fim de que sejam tomadas ações de acordo com um conjunto de regras e procedimentos. Como uma definição alternativa, tem-se que *workflow* é uma representação específica de um processo, a qual é desenvolvida de modo que os mecanismos formais de coordenação entre atividades, aplicações e participantes do processo possam ser controlados por um sistema de informação, o sistema de gerenciamento de *workflow* (MUEHLEN, 2004).

Dentre as áreas de aplicação de *workflow* incluem-se aquelas com base na ordenação e controle de atividades as quais podem ser automatizadas, tais como processos industriais, manufatura, processos jurídicos, ensino à distância e controle de licitações, entre outros. A gestão por processos associada à tecnologia de workflow pode trazer diversos benefícios à organização, tais como: (a) descrição precisa e não ambígua dos processos de negócio existentes; (b) melhoria na definição de novos processos; (c) maior eficácia na coordenação do trabalho entre diferentes agentes; (d) obtenção, em tempo real, de informações precisas sobre o andamento dos processos e; (e) padronização dos processos executados, de forma manual ou automatizada, pela organização (THOM, 2007a).

Muitas vezes, diferentes passos de um processo de negócio podem estar sendo executados redundantemente ou de forma ineficiente. A tecnologia de *workflow*, através da automação dos processos de negócio, contribui para a redução de custos, tempo de execução, erros, bem como redundâncias na execução dos processos. Ao mesmo tempo, ela aumenta o controle sobre eles levando ao incremento da qualidade dos processos, dos seus resultados e da organização como um todo (IOCHPE, 2001), (MUEHLEN, 2004), (THOM, 2007b).

Workflow está intimamente associado à reengenharia de processos de negócio, preocupando-se com a análise, modelagem, definição e subsequente implementação do núcleo de um processo de negócio de uma organização, ou outra entidade de negócios (LOPES, 2003). Apesar de nem todas as atividades de reengenharia de processos resultarem em implementações de *workflow*, tal tecnologia é muito apropriada, pois permite uma separação entre a lógica do processo de negócio e seu suporte operacional, facilitando a incorporação de mudanças futuras nas regras procedurais que definem os processos de negócio. Alternativamente, nem todo *workflow* é fruto de uma reengenharia de processos, visto que pode ser resultado de uma implementação automatizada de um processo de negócio pré-existente (WORKFLOW MANAGEMENT COALITION, 1995).

Os **Sistemas de gerenciamento de *workflow*** (WfMS – *Workflow Management System*) são sistemas de *software* que interpretam a definição formal de um processo de negócio, a qual é especificada através de uma linguagem de execução de processos de negócio (OWEN, 2003). Além disso, controlam o andamento deste de acordo com a sua definição, interagem com os participantes e invocam os aplicativos externos. São eles os responsáveis pela definição, criação e gerenciamento da execução do *workflow*, fornecendo, inclusive, funções administrativas e de supervisão para permitir a re-atribuição e escalonamento de tarefas, realização de auditorias e gerenciamento de informações (WORKFLOW MANAGEMENT COALITION, 1998a), (WORKFLOW MANAGEMENT COALITION, 1998b). Com o desenvolvimento deste tipo de aplicativo ao longo dos últimos anos, não mais apenas o trabalho pode ser realizado automaticamente pelos computadores, mas a sua gerência também (PLESUMS, 2002). Dentre alguns sistemas existentes, pode-se citar Staffware, FileNet Visual WorkFlo, MQ Series Workflow, SAP R/3 Workflow, Fujitsu´s i-Flow, Verve, Forte Conductor, HP Changengine, entre outros (KIEPUSZEWSKI, 2003).

Os sistemas de gerenciamento de *workflow* estão para a lógica dos processos assim como os sistemas de gerenciamento de banco de dados (SGBDs) estão para os dados em relação à aplicação. Como se observa na Figura 2.1, enquanto estes separam o gerenciamento dos dados dos problemas relativos à aplicação, aqueles isolam a lógica dos processos da lógica da aplicação (SCHOOL OF INFORMATION TECHNOLOGY

AND ELECTRICAL ENGINEERING, 2002). De certo modo, os sistemas de gerenciamento de *workflow* possuem alguma semelhança com sistemas de banco de dados, no fato de monitorar o estado de um sistema e disparar atividades conforme determinados eventos (MUEHLEN, 2004).

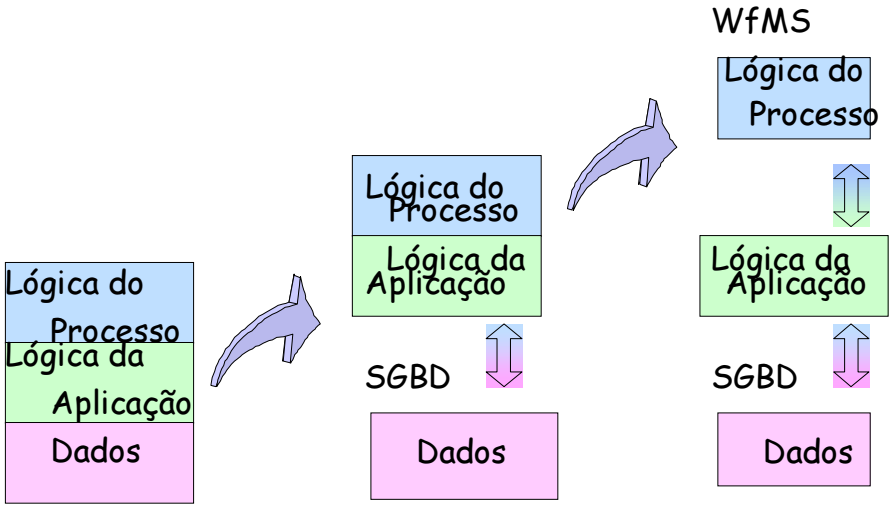

Figura 2.1: Sistemas de gerenciamento de banco de dados e sistemas de gerenciamento de workflow

Um exemplo de processo de negócio passível de ser automatizado através de um sistema de *workflow* é a análise de solicitação de verbas em uma empresa. Sem a utilização da tecnologia, cada responsável pela aprovação da solicitação deve receber uma cópia desta e enviar um documento contendo o seu parecer. Por outro lado, com o uso de um sistema de *workflow*, tal análise pode-se dar de forma muito mais econômica e eficiente na medida em que cada responsável pela decisão recebe uma cópia digital da requisição e responde ao solicitante, também de forma digital. Deste modo, tem-se uma economia no tempo de execução da tarefa, que tem uma resposta muito mais rápida, podendo inclusive atingir distâncias muito maiores, já que documentos digitais viajam a grandes distâncias de forma muito mais rápida do que correspondências ou documentos físicos. Além disto, há uma economia em termos de gastos financeiros, já que diversas cópias em papel da solicitação são substituídas por documentos digitais. A tecnologia não exige que os analistas estejam próximos do solicitante, permitindo uma distribuição do trabalho de forma eficiente e transparente. Como uma extensão do correio eletrônico, a tecnologia de *workflow* permite rápida comunicação e colaboração entre usuários geograficamente dispersos.

Para que um processo de negócio possa ser manipulado e executado por um sistema de gerenciamento de *workflow*, este deve possuir uma **definição de processo**, isto é, uma ordem parcial de atividades e relacionamentos entre elas, critérios para estabelecer início e término das atividades, bem como informações sobre os participantes, aplicações externas e dados utilizados e gerados durante a execução do processo. Em outras palavras, uma representação formal do que deve acontecer, traduzindo um processo do mundo real com o uso de métodos de análise, modelagem e definição em uma forma computadorizada capaz de ser interpretada por um sistema gerenciador de *workflow*. Sub-processos também podem ter uma definição separada e serem referenciados dentro de uma definição de um processo maior, possibilitando o reuso de componentes e modularização de projeto (WORKFLOW MANAGEMENT COALITION, 1998a).

Uma definição de processo é composta por **atividades**, as quais podem ser definidas como um fragmento de trabalho o qual constitui um único passo lógico dentro de um processo. Tipicamente, uma atividade é a menor unidade de trabalho que é escalonada por um sistema de gerenciamento de *workflow* ao longo da realização de um processo. Pode ser uma **atividade manual**, a qual, portanto, não suporta automação, ou uma **atividade automatizada** capaz de ser controlada por um computador utilizando um sistema de gerenciamento de *workflow*. Uma atividade automatizada pode requerer recursos humanos ou mecânicos para a realização do processo, são os chamados **participantes do *workflow***. Quando um humano é necessário, a atividade é alocada a este participante e inserida em sua lista de trabalho. Quando o recurso mecânico é utilizado, uma aplicação pode ser automaticamente chamada (WORKFLOW MANAGEMENT COALITION, 1999a). Uma atividade manual pode perfeitamente fazer parte de um processo de negócio e ser incluída em sua definição de processo, entretanto, esta não faz parte do *workflow* implementado para a execução deste processo suportada por computador já que, como o próprio nome indica, é uma atividade a qual deve ser realizada manualmente, não suportando o auxílio de máquinas.

Em um fluxo de trabalho de um processo de negócio temos os *gateways*, os quais são responsáveis pelo controle deste fluxo. Existem diferentes tipos de controle de fluxo, também conhecidos como roteamento (THOM, 2006a). A Tablea 2.1 tem estes diferentes tipos de roteamento esclarecidos.

Tabela 2.1: Tipos de roteamento

Roteamento	**Descrição**
Sequencial	Uma atividade só é habilitada após a conclusão de uma outra atividade no mesmo processo.
AND-split	É uma única *thread* de controle se dividindo em múltiplas outras, as quais podem ser executadas em paralelo, permitindo que atividades sejam executadas simultaneamente ou em qualquer ordem.
AND-join	Também conhecido como sincronização, é o ponto do *workflow* onde duas ou mais atividades paralelas em execução convergem para um único fluxo de execução de controle comum.
OR-split	Refere-se a um ponto do *workflow* onde uma única *thread* de controle decide por quais dentre vários caminhos de execução continuar o fluxo.
OR-join	Compreende um ponto dentro do *workflow* onde duas ou mais atividades paralelas re-encontram-se numa única atividade como próximo passo do *workflow*.
XOR-split	Conhecido como roteamento condicional ou decisão, é um ponto do processo do *workflow* onde, baseado em uma decisão ou em dados de controle, decide por um dos vários caminhos possíveis.
XOR-join	é o ponto do processo de *workflow* onde dois ou mais ramos alternativos encontram-se sem sincronização. Neste controle de fluxo nenhum dos ramos alternativos é executado em paralelo.

A utilização de um sistema de gerenciamento de *workflow* dentro de uma organização traz inúmeras vantagens que tornam tanto clientes como funcionários mais satisfeitos (PLESUMS, 2002) (ALLEN, 2001). A saber:

- cada participante recebe automaticamente suas tarefas de acordo com suas atribuições, não sendo mais necessário que uma pessoa realize esta distribuição, evitando-se, assim, que pessoas erradas recebam o trabalho errado;
- todo o procedimento é formalmente documentado, garantindo sua realização exatamente como planejado;
- o participante mais adequado pode ser encontrado de forma rápida e automática para a realização de uma tarefa específica através de seus atributos, melhorando de forma geral a eficiência e qualidade do processo e reduzindo custos;
- os funcionários sentem-se melhor ao saber exatamente o que devem realizar;
- o processo é mais bem controlado e facilmente gerenciável;
- a economia de custos é atingida, entre outros fatores menos diretos, como a redução de pessoal, já que o trabalho é automaticamente distribuído e menos treinamento é necessário, pois o trabalho é passado de uma pessoa a outra, a qual se concentra em suas tarefas, sem a necessidade de conhecer a organização inteira;
- pode surgir a oportunidade de melhorias na execução de um passo de um processo, visto que para a automação cada etapa deve ser analisada e documentada, entre outras vantagens.

Como comentado anteriormente, para que os processos possam ser executados sobre um gerenciador de *workflow*, estes devem possuir uma definição formal. Esta se da por modelos construídos com base em alguma notação para que então possam ser incorporados ao gerenciador. Atualmente, existe uma grande variedade de notações para a modelagem de processos de negócio, tais como BPMN (OBJECT MANAGEMENT GROUP, 2006), EPC (MENDLING, 2005), UML (LARMAN, 2000), entre outras.

2.2 Padrões de Modelagem

Dentro da modelagem de processos, seja com BPMN ou outras notações, tem-se notado o aparecimento de certas estruturas recorrentes, isto é, padrões (ex.: solicitação de execução de tarefa, notificação, decisão) (THOM, 2006a). Um **padrão** é a abstração de uma forma concreta a qual se mantém recorrente em contextos específicos não arbitrários (GAMMA apud THOM, 2006b).

Padrões ajudam a promover boas práticas de desenvolvimento. Eles têm sido usados em diferentes domínios, desde organizações e processos, até ensino e arquitetura. Contudo, padrões para processos de negócio, assim como para modelagem de processos em *workflow* ainda são temas de discussão e pesquisa (SCHOOL OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING, 2002).

Sabe-se que a identificação e utilização de padrões em *workflow* pode ser aplicada para melhorar a qualidade e desempenho da fase de desenvolvimento dos modelos dos processos (THOM, 2006b). Em vista disto, numerosas pesquisas neste campo de estruturas reutilizáveis têm se desenvolvido buscando identificar e difundir construções que aparecem frequentemente nos processos modelados (AALST, 2003), (RUSSEL, 2004a), (RUSSEL, 2004b), (BRADSHAW, 2005), (RUSSEL, 2006), (THOM, 2006a),

(THOM, 2009). As abordagens mais expressivas são no campo dos padrões de controle e dados (AALST, 2003), assim como nos padrões orientados a recursos e aplicações (RUSSEL, 2004). Os primeiros buscam capturar os vários modos como os dados são representados e utilizados em *workflow*, enquanto que os últimos preocupam-se com os recursos. Existem também abordagens relacionando o desenvolvimento do *workflow* com um conjunto de "partes" de processos de negócio recorrentes, como a solicitação de execução de uma atividade, uma notificação ou uma atividade financeira, ou "partes" que precisam ser executadas atomicamente pelo processo do *workflow* (THOM, 2006a).

Recentemente uma variedade de padrões de *workflow* foi sugerida para capturar diferentes aspectos em sistemas PAIS (*Process Aware Information Systems*) incluindo fluxo de controle e de dados, recursos, mudança de processo e tratamento de exceções (AALST, 2003), (WEBER, 2007).

Com relação ao suporte a padrões por parte das ferramentas de modelagem, a YAWL é uma ferramenta a qual suporta padrões de *workflow*. No entanto, ela utiliza redes estendidas de *workflow* como blocos de construção para especificações de *workflows*. Múltiplas redes estendidas de *workflow* envolvidas em uma especificação podem ser conectadas umas com as outras através da composição de tarefas (AALST, 2005a). A abordagem PICTURE propõe um conjunto de 37 blocos de construção específicos para um domínio. Mais precisamente, estes blocos de construção são utilizados por usuários finais na Administração Pública para capturar uma visão geral de processos (BECKER, 2007). Já o ProCycle é uma outra abordagem que vem implementando padrões de mudança de processos em ADEPT2 (WEBER, 2009).

Apesar das alternativas apresentadas, as ferramentas atuais de modelagem de *workflow* tais como Intalio (INTALIO, 2006), ARIS Toolset e WBI Modeler não oferecem ao usuário meios acessíveis para definir, consultar e reutilizar **padrões de atividade** como blocos de construção para a modelagem de processos de negócio. Um dos maiores problemas é a inexistência de um mapeamento entre os padrões de atividades orientados a objetivos, como padrões representando a solicitação da execução de uma atividade, uma notificação, uma decisão ou uma aprovação, por exemplo, para (meta) modelos de processos e ferramentas de modelagem de processos respectivamente. A Sub-Seção a seguir apresenta em maiores detalhes o conceito de padrões de atividade conforme Thom (2006b).

2.2.1 Padrões de Atividade

Conjuntos de atividades podem ser modelados como **blocos de atividades**. Quando dentro de um bloco, as atividades devem ter sua execução iniciada pela primeira e as atividades subsequentes devem ser executadas seguindo a ordem parcial estabelecida pelas transições até que seja atingida uma atividade de saída. A execução do *workflow* retorna, então, para a atividade que segue o bloco (THOM, 2006a). No contexto de um processo de negócio, assim como em um processo de *workflow*, existe uma variedade de "partes" de processos, as quais podem ser entendidas como blocos de atividades auto-contidos com uma semântica específica e bem definida. Convém observar que a mesma "parte" pode ser repetida dentro do mesmo processo. Durante a execução, as diferentes cópias da mesma "parte" podem receber como parâmetros os mesmos ou outros valores.

Os padrões desenvolvidos em Thom (2006b) representam partes recorrentes de processos de negócio (como a aprovação de um documento, por exemplo) as quais precisam ser atomicamente executadas dentro de um processo de *workflow*, diferindo

dos padrões que representam fluxos de controle e dados fortemente pesquisados em Aalst (2003) e Russel (2004a). Esses padrões estão mais próximos de macro atividades que são executadas dentro de um processo atingindo, assim, um nível de granularidade mais próximo do vocabulário do nível de abstração que os especialistas do domínio utilizam para descrever um processo. Desta maneira, eles são capazes de tornar mais fácil a tarefa de análise e modelagem dos mesmos. Por estarem representando atividades, estes padrões são conhecidos como **padrões de atividade**.

Cada padrão em Thom (2006b) é representado por uma atividade de bloco, o que significa que todas as sub-atividades dentro do bloco devem ser completadas antes que o *workflow* dentro do qual está inserido o bloco de atividade em questão possa continuar sua execução. O conceito de bloco de atividades é aplicado com o intuito de facilitar a implementação dos padrões na tecnologia atual existente, como através da utilização do *Oracle Workflow Builder*, por exemplo, e para encapsular a sua semântica representando suas atividades como atômicas. Os fragmentos de processo recorrentes que são reconhecidos como padrões de atividade são: *Solicitação de execução de atividade sem resposta* (*Performativo Unidirecional*), *solicitação de execução de atividade com resposta* (*Performativo Bidirecional*), *aprovação*, *notificação*, *tomada de decisão*, *solicitação de informação*, *Q&A* e *atividade financeira* (THOM, 2006b), (THOM, 2009).

Como alguns padrões podem requerer a entrada e/ou saída de alguns parâmetros e o conceito de atividade de bloco que é utilizado para representá-los não suporta parâmetros, é utilizada a perspectiva de transação “emprestada” da teoria da serializabilidade. Desta maneira, um parâmetro de entrada é representado como uma operação única de leitura de um banco de dados e um parâmetro de saída é representado como uma operação de escrita em um banco de dados (THOM, 2006b).

Figura 2.2: Processo de lançamento de novos produtos

A Figura 2.2 ilustra um processo onde podem ser identificados alguns padrões, ou seja, alguns fragmentos que podem ser entendidos como blocos de atividade autocontidos com uma específica e bem definida semântica. Cada um destes fragmentos pode aparecer diversas vezes em uma definição de processo e durante o tempo de execução diferentes cópias lógicas do mesmo fragmento podem possuir diferentes valores como parâmetros. O processo da Figura 2.2 primeiro notifica as partes envolvidas sobre o lançamento de um novo produto, avalia o lançamento, notifica o departamento financeiro sobre a aprovação e checa se existem novos lançamentos para avaliação. Estas atividades representam fragmentos que estão relacionados com padrões de notificação, aprovação e solicitação de execução de uma atividade, como pode ser observado nas descrições das atividades sendo realizadas.

Estes padrões, assim como outros, como o de decisão, por exemplo, são bastante recorrentes no desenho de *workflows* e precisam ser re-desenhados a cada nova modelagem. Isto significa que diversas cópias lógicas do mesmo fragmento de processo podem ser utilizadas com os mesmos ou diferentes parâmetros, como uma aprovação

por um único ou múltiplos autores, por exemplo. Apesar de possuírem uma semântica bem característica, existem por enquanto poucas pesquisas relacionando este tipo de estrutura de processos com padrões de *workflow* (THOM, 2006b). A Tabela 2.2 apresenta os padrões desenvolvidos em Thom (2006b; e, 2009) juntamente com a sua semântica.

Tabela 2.2: Padrões de atividade e sua semântica

Padrão	**Semântica**
Performativo Unidirecional	Um remetente solicita a execução de uma tarefa particular a um outro participante do processo. O remetente continua a execução do processo imediatamente após ter enviado a solicitação de execução de atividade.
Performativo Bi-direcional	Um remetente solicita a execução de uma tarefa particular a um outro participante do processo. O remetente aguarda até que este participante o notifique que a tarefa foi realizada.
Notificação	O estado ou resultado da execução de uma atividade é comunicado a um ou mais participantes do processo.
Aprovação	Um objeto (um documento, por exemplo) precisa ser aprovado por um ou mais papéis organizacionais. Dependendo do respectivo contexto, a avaliação é executada uma única vez (aprovação única), ou pode ser em sequência (aprovação iterativa), ou pode ser em paralelo (aprovação concorrente).
Tomada de Decisão	A execução de uma ou múltiplas atividades é solicitada. Dependendo dos resultados das execuções das atividades solicitadas o processo continua sua execução por uma ou mais ramificações. Mais precisamente, este padrão permite incluir uma atividade de decisão com conectores para diferentes ramos de execução, cada um deles associado a uma condição específica de transição. Serão selecionados para execução somente aqueles ramos cujas condições de transição tornem-se verdadeiras com a execução da atividade de decisão.
Solicitação de Informação	Um ator solicita certa informação a um participante do processo. Ele continua a execução do processo assim que recebe a informação.
Q&A	Permite que um ator de um processo formule uma questão. Este padrão ainda contempla a identificação do papel adequado a responder a questão, envio desta para este papel e aguardo da resposta. Como generalização, esta questão pode ser enviada para múltiplos papéis resultando em múltiplas respostas.

Padrão Financeiro	Representa um processo financeiro que manipula e, eventualmente, gera um valor monetário.

Como um exemplo de padrão de atividade tem-se a Figura 2.3. Ela ilustra o *padrão performativo unidirecional*. Este padrão representa uma mensagem unidirecional como descrito em zur Muehlen (2004). Um remetente utiliza mensagens performativas unidirecionais para solicitar a execução de uma atividade particular a um receptor (humano ou máquina) envolvido no processo. O remetente continua a execução de sua parte do processo imediatamente após ter enviado a solicitação. Por exemplo, em um processo de aquisição, a execução de uma atividade para cancelar parcialmente um pedido pode ser solicitada por um gerente se alguma irregularidade ocorre. O fluxo continua assim que a atividade de cancelamento é solicitada.

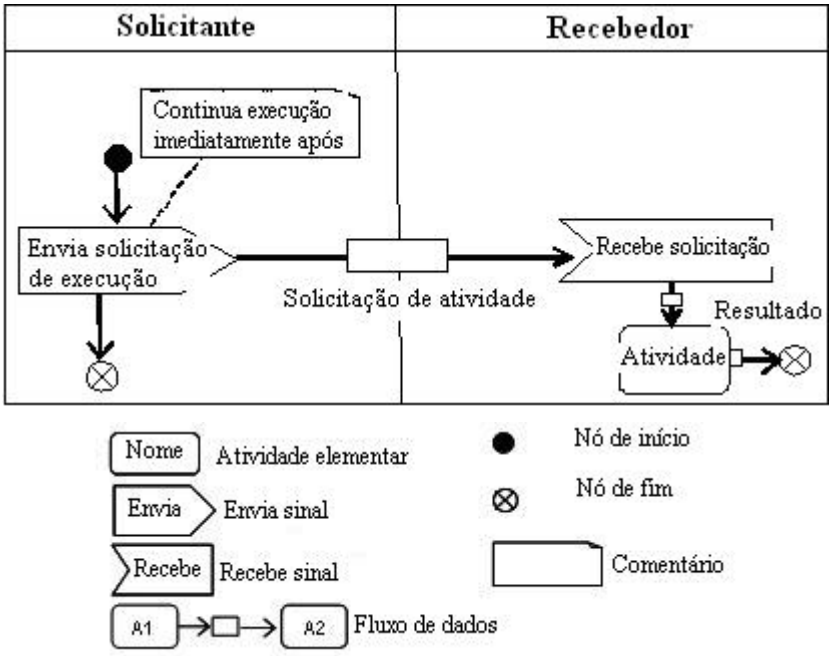

Figura 2.3: Padrão performativo unidirecional

Diferentemente dos padrões propostos por Aalst (2003), os quais focam mais em aspectos comportamentais em termos do controle de fluxo, dados e recursos, Thom enfatiza diferentes tipos de atividades de bloco baseadas no conceito da WfMC. Comparando-se estes padrões com aqueles da SAP, MIT e WIDE, pode-se dizer que estes são mais independentes da aplicação, enquanto que aqueles focam em domínios mais específicos (THOM, 2006b).

Os padrões apresentados em Thom (2006b) foram derivados principalmente de um estudo da literatura sobre aspectos estruturais organizacionais, como centralização na tomada de decisão, e de tipos de (sub)processos de negócio. Os diferentes padrões são compostos para gerar um modelo de processo utilizando-se as estruturas de roteamento apresentadas na Tabela 2.1. Através de um estudo empírico, no qual 190 processos reais foram analisados, a existência dos padrões de atividade apresentados foi confirmada (THOM, 2008a), (THOM, 2008b), (THOM, 2009). Nesta abordagem foi demonstrado que os modelos de processos analisados podem ser completamente desenhados com base nos padrões apresentados na Tabela 2.2, isto é, o conjunto de padrões identificados mostrou-se suficiente e necessário para modelar os 190 processos em um certo nível de granularidade pelo menos.

3 PROCESSO DE DESCOBERTA DE CONHECIMENTO

Seguindo a apresentação dos conteúdos chave para o completo desenvolvimento deste trabalho, este Capítulo dará uma introdução ao processo de descoberta de conhecimento e seus principais conceitos e etapas.

A tecnologia de *Knowledge Discovery in Databases* (KDD – Descoberta de Conhecimento em Bancos de Dados) abrange teorias e ferramentas para auxiliar os humanos a extrair conhecimento a partir de grandes e crescentes volumes de dados digitais (FAYYAD, 1996). KDD é um campo que está preocupado com o desenvolvimento de técnicas e métodos para o mapeamento de dados em baixo nível - os quais são, geralmente, muito volumosos para um completo entendimento e compreensão – para outras formas que sejam mais compactas como, por exemplo, um pequeno relatório; mais abstratas como, por exemplo, uma aproximação descritiva ou um modelo do processo que gerou os dados; ou mais úteis como, por exemplo, um modelo preditivo para estimar valores de futuros casos.

No núcleo do processo de KDD está a aplicação de métodos específicos de mineração de dados para a descoberta e extração de padrões (FAYYAD, 1996). Esta etapa também será abordada em maiores detalhes neste Capítulo.

3.1 Descoberta de Conhecimento e Mineração de Dados

Segundo Fayyad (1996), KDD refere-se ao processo de descoberta de conhecimento útil a partir de volumes de dados como um todo, enquanto que mineração de dados refere-se a um passo particular deste processo.

Historicamente, a noção de encontrar padrões úteis em massas de dados tem recebido nomes como mineração de dados, extração de conhecimento, descoberta de informação, colheita de informação, *data archaeology*, processamento de padrões de dados, entre ouros (FAYYAD, 1996).

O termo **mineração de dados** foi o que ganhou maior popularidade no campo de bancos de dados. Em 1989 foi cunhado o termo **descoberta de conhecimento em bancos de dados** para enfatizar que conhecimento é o produto final de um processo de descoberta dirigido pelos dados dando origem à sigla KDD (PIATETSKY-SHAPIRO apud FAYYAD, 1996).

A mineração de dados é a aplicação de algoritmos específicos para a extração de padrões a partir dos dados. Além da mineração de dados, KDD possui etapas para preparação, seleção e limpeza de dados, incorporação de conhecimento inicial e interpretação dos resultados da mineração, os quais são essenciais para garantir que um conhecimento utilizável seja extraído.

As Sub-Seções seguintes darão mais detalhes com relação ao processo de descoberta de conhecimento e a sua etapa de mineração de dados, respectivamente.

3.1.1 Descoberta de Conhecimento (KDD)

KDD é o processo iterativo e interativo, não trivial de identificar padrões válidos, potencialmente úteis e inteligíveis a partir de massas de dados (FAYYAD, 1996). Este processo envolve diversas etapas e muitas decisões tomadas pelo usuário.

Nesta definição, dados são um conjunto de fatos, como casos em um banco de dados e padrões são expressões em alguma linguagem descrevendo um subconjunto dos dados ou um modelo aplicável ao subconjunto. Portanto, encontrar um padrão pode também ser visto como encaixar um modelo nos dados, encontrar uma estrutura para eles ou, de forma geral, fazer qualquer descrição em alto nível de um conjunto de dados. O termo processo implica que KDD compreende um conjunto de passos, os quais são repetidos em múltiplas iterações. Por *não trivial* entende-se que algum tipo de busca ou inferência é necessário, ou seja, não é uma computação direta de quantidades pré-definidas como no cálculo de uma média de um conjunto de valores.

Para exemplificar alguns conceitos do processo de KDD considere-se o problema de dizer se uma partida de tênis acontecerá dadas as condições do tempo. Neste exemplo, os dados são as informações das partidas que já aconteceram, isto é, as condições climáticas como temperatura, vento e umidade e se houve jogo ou não. Padrões são informações que são extraídas após o processo de KDD ser aplicado a estes dados e podem ser algo do tipo: “se o dia está quente, com vento moderado e úmido, então há jogo”. Diversos padrões como este compõem um modelo que descreve os dados e podem ser utilizados para fazer previsões sobre partidas futuras, por exemplo.

É importante que os padrões descobertos no processo de KDD sejam válidos em novos dados com certo grau de certeza. Além disso, deseja-se que eles sejam fáceis de serem entendidos (pelo sistema e preferencialmente pelo usuário) e capazes de trazer algum benefício ao usuário ou tarefa. Isto exige que sejam definidas medidas quantitativas para avaliar os padrões extraídos. Em alguns casos é fácil estabelecer medidas de certeza, tais como a acurácia da predição estimada em novos dados, por exemplo. Noções de inteligibilidade são mais subjetivas. Em certos contextos, esta pode ser estimada através da simplicidade, como o número de bits necessários para descrever um padrão, por exemplo. Uma noção importante é o valor de interesse do padrão, o qual é uma medida que engloba valores de novidade, valor, simplicidade, validade e utilidade (FAYYAD, 1996).

Dadas estas noções, um padrão é considerado conhecimento se ele excede um certo limiar de interesse. Deste modo, o conhecimento é puramente orientado a um domínio específico e determinado pela função de interesse utilizada.

Todo o processo de KDD pode ser visto como composto de nove diferentes passos, os quais podem ser encaixados nas diferentes etapas ilustradas na Figura 3.1:

1. Desenvolvimento e *entendimento do domínio de aplicação* e conhecimento especifico anterior relevante e identificação do objetivo do processo de KDD do ponto de vista do consumidor.
2. *Seleção* de um conjunto de dados alvo onde a descoberta deve ser realizada.

3. Limpeza e *pré-processamento* dos dados. Inclui operações como remoção de ruídos, se apropriado, decisão de estratégias para lidar com dados faltantes, informações sequenciais e mudanças conhecidas.
4. Redução e projeção dos dados, encontrando meios para *transformar* e representar os dados conforme o objetivo da tarefa.
5. Casar os objetivos do processo de KDD a um *método particular de mineração de dados*, como sumarização, classificação, regressão, clusterização e assim por diante.
6. Análise exploratória e seleção de modelo e hipótese, escolhendo os *algoritmos de mineração de dados* e os métodos para busca por padrões.
7. *Mineração de dados* buscando por padrões de interesse em uma forma de representação particular, ou um conjunto de representações, incluindo regras de classificação ou árvores, por exemplo.
8. *Interpretação dos padrões minerados*, possivelmente retornando a alguns dos passos anteriores para mais iterações. Este passo também pode incluir a visualização dos padrões e modelos extraídos ou visualização dos dados conforme os modelos extraídos.
9. Utilização do *conhecimento extraído*, incorporando-o a algum sistema existente para posteriores interações, ou simplesmente documentando-o e repassando às partes interessadas. Também é realizada uma checagem e resolução de conflitos sobre o conhecimento extraído comparando-o com conhecimentos antecedentes.

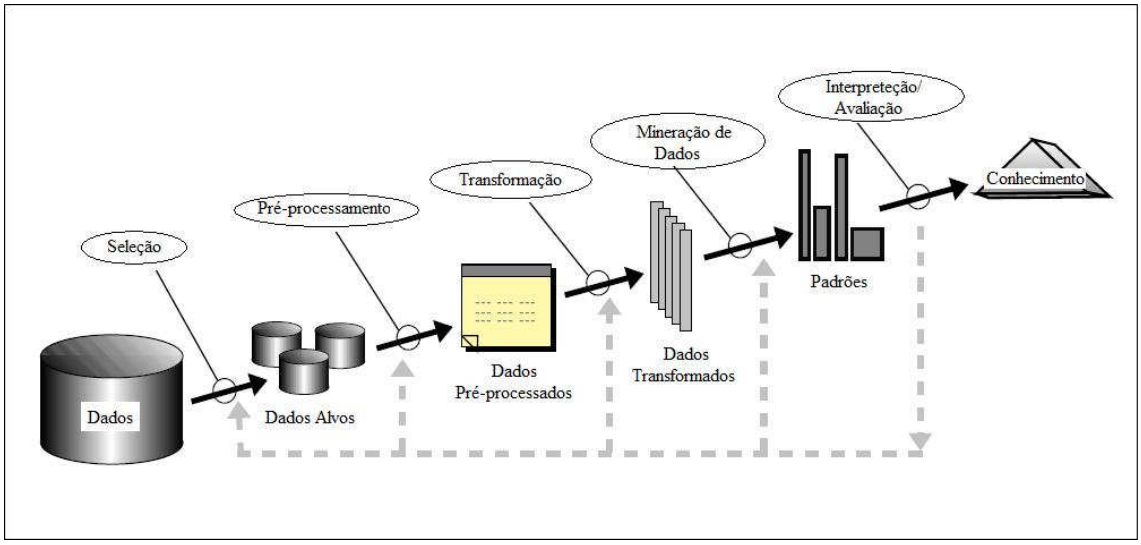

Figura 3.1: Etapas de KDD (FAYYAD, 1996)

A simples e cega aplicação de um processo de KDD pode facilmente levar a descoberta de padrões sem sentido e inválidos e, por este motivo, os seus objetivos devem ser definidos conforme o uso pretendido dos resultados. Estes objetivos podem ser divididos em duas classes: (1) verificação e (2) descoberta. Através da verificação o sistema implementado para resolver o problema é limitado a verificar a hipótese do usuário. Na descoberta o sistema encontra novos padrões de forma autônoma. O objetivo de descoberta, por sua vez, ainda pode ser divido em duas classes: (I) predição, onde o sistema encontra padrões para prever o comportamento futuro de certas

entidades, como no exemplo das partidas de tênis; e (II) descrição, onde o sistema encontra padrões para apresentar ao usuário em uma forma humanamente inteligível, como por exemplo, agrupar um conjunto de pessoas dados seus interesses musicais. Apesar dos limites entre predição e descrição não serem rígidos, a distinção é útil para a compreensão do objetivo da descoberta como um todo (HAN, 2001).

Os objetivos da predição e descrição podem ser atingidos utilizando-se uma série de métodos. Um método bastante conhecido é a **classificação**, a qual envolve a descoberta de uma função capaz de mapear (classificar) um item de dado em uma dentre várias classes pré-definidas. Exemplos da utilização de classificação como parte de aplicações de descoberta de conhecimento podem ser vistos em sistemas de classificação de tendências de mercados financeiros, ou identificação automática de objetos de interesse em grandes bancos de dados de imagens (FAYYAD, 1996).

A **regressão** é um método de descoberta de uma função que mapeia um item de dados para uma variável real de predição. Exemplos da aplicação de regressão podem ser vistos na predição da quantidade de biomassa existente em uma floresta dadas as medições remotas de microondas, estimação da probabilidade de sobrevivência de um paciente dados os resultados de um conjunto de testes de diagnósticos, predição da demanda por um produto novo como uma função do investimento em anúncios, entre outros (FAYYAD, 1996).

Clusterização é um método descritivo comum onde se procura identificar um conjunto finito de categorias, ou *clusters*, para descrever os dados. As categorias podem ser mutuamente exclusivas e exaustivas, ou consistir de uma representação mais rica, incluindo hierarquias ou categorias sobrepostas. Exemplos da aplicação deste método podem ser vistos na descoberta de subpopulações homogêneas de consumidores em bancos de dados de mercados (HAN, 2001).

Já a **sumarização** envolve meios para encontrar uma descrição compacta de um subconjunto dos dados. Um exemplo simples seria o cálculo da média e desvio padrão dos dados. Métodos mais sofisticados incluem a derivação de regras de sumarização (AGRAWAL, 1996), técnicas de visualização multivaloradas e a descoberta de relacionamentos funcionais entre as variáveis (ZEMBOWICZ apud AGRAWAL, 1996). Técnicas de sumarização são normalmente aplicadas na análise exploratória interativa de dados e na geração automática de relatórios.

A **modelagem de dependências** é um outro método que consiste em encontrar um modelo que descreva dependências significantes entre as variáveis. Os modelos de dependência existem em dois níveis: (1) o nível estrutural, onde são especificadas quais variáveis são localmente dependentes entre si; e (2) o nível quantitativo que especifica as forças das dependências utilizando alguma escala numérica. Este método encontra aplicações no desenvolvimento de sistemas especialistas probabilísticos do ramo médico, recuperação de informações e modelagem do genoma humano (FAYYAD, 1996).

Por fim, a **detecção de mudanças e desvios** é um método que se foca em descobrir as mudanças mais significativas dos dados a partir de valores anteriormente medidos. Como exemplo de aplicação desta técnica pode-se citar aplicações de detecções de fraudes em transações bancárias (FAYYAD, 1996).

3.1.2 Mineração de dados dentro de KDD

Como apresentado, a mineração de dados é um componente da KDD que envolve a aplicação iterativa de técnicas de mineração de dados particulares. Esta etapa apoia-se fortemente em conhecidas técnicas oriundas do aprendizado de máquinas, reconhecimento de padrões e estatística para encontrar padrões nos dados.

A mineração de dados consiste na aplicação de análise de dados e algoritmos de descoberta que, sob limitações aceitáveis de eficiência computacional, produz uma enumeração particular de padrões ou modelos sobre os dados. Neste contexto, o espaço de padrões é normalmente infinito e restrições computacionais práticas impõem limites no subespaço que é explorado pelos algoritmos de mineração de dados (FAYYAD, 1996).

Os métodos para resolução de problemas de KDD apresentados na Sub-Seção 3.1.1 são como linhas gerais para definir como deve funcionar a mineração de dados. Estas ideias são implementadas em algoritmos específicos que realizam a tarefa de minerar os dados. Um algoritmo deste tipo possui três partes: representação do modelo, avaliação do modelo e busca (FAYYAD, 1996).

A representação do modelo é a linguagem utilizada para descrever padrões capazes de serem descobertos. Se a representação é limitada, os resultados são menos precisos. Por outro lado, se a representação é muito detalhada, aumenta-se o risco de particionar demais os dados de treino, resultando em menor precisão na predição de novos dados.

Para a avaliação de um modelo são utilizados critérios de avaliação. Estes critérios são declarações quantitativas do quanto um padrão está de acordo com os objetivos do processo de KDD. Por exemplo, modelos preditivos podem ser avaliados através da precisão da predição empírica em algum conjunto de testes. Modelos descritivos podem ser avaliados através da precisão preditiva, utilidade e inteligibilidade do modelo resultante.

O método de busca de um algoritmo de mineração de dados consiste de dois componentes: busca de parâmetros e busca de modelos. Estando fixados a representação do modelo (ou família de representações) e os critérios de avaliação, o algoritmo de mineração de dados deve encontrar parâmetros e modelos para a família selecionada que otimizem o critério de avaliação (FAYYAD, 1996). A seguir são comentados alguns métodos de mineração de dados no contexto de representação do modelo, avaliação e busca.

As **árvores de decisão** e **regras** que utilizam divisões invariáveis são métodos de mineração de dados que possuem uma simples forma representacional, tornando o modelo inferido relativamente fácil de ser compreendido pelo usuário. Contudo, a restrição a uma representação em árvore ou regra particular pode restringir significativamente a forma funcional do modelo e, portanto, o seu poder de aproximação. Se o espaço do modelo é ampliado para permitir expressões mais gerais, então o modelo é mais poderoso na predição, mas pode ser mais difícil de ser compreendido. Métodos de busca gulosa, que envolvem o crescimento e poda de regras e estruturas de árvores, são tipicamente utilizados para explorar o espaço exponencial de modelos possíveis. Árvores e regras são utilizadas primeiramente para modelagens preditivas, tanto para classificação quanto para regressão (FAYYAD, 1996).

Métodos de classificação e regressão linear consistem de uma família de técnicas para predição que encaixam combinações lineares e não-lineares de funções base, como

sigmoides, *splines* e polinômios, a combinações das variáveis de entrada. Exemplos incluem **redes neurais de retro-alimentação** e **métodos de *splines* adaptativas**. Métodos de regressão linear, apesar de possuírem grande poder de expressão, podem ser difíceis de serem interpretados (FAYYAD, 1996).

Métodos baseados em exemplos possuem uma representação extremamente simples: são os próprios exemplos do banco de dados que aproximam um modelo. Predições de novos valores são derivadas das propriedades de exemplos similares no modelo cuja predição é conhecida. Técnicas para a realização desta tarefa incluem classificação pelo vizinho mais próximo e sistemas de raciocínio baseados em casos (KOLODNER apud AGRAWAL, 1996). Uma desvantagem deste método em comparação com aqueles baseados em árvores é a necessidade de uma métrica bem definida para calcular a distância entre os dados. A avaliação do modelo é tipicamente baseada em estimativas de validações cruzadas do erro de predição. Parâmetros do modelo para serem estimados incluem o número de vizinhos utilizados na predição e a própria métrica de distância. Assim como nos métodos de regressão não-linear, métodos baseados em exemplos são poderosos em termos das propriedades de aproximação, mas podem ser difíceis de serem interpretados, porque o modelo está implícito nos dados e não explicitamente formulado (FAYYAD, 1996).

Os **modelos gráficos de dependências probabilísticas** especificam dependências através de uma estrutura de grafo. Na sua forma mais simples, o modelo especifica quais variáveis são diretamente dependentes de cada uma. Tipicamente são utilizados com variáveis categóricas ou de valores discretos, mas extensões para casos especiais, como densidades Gaussianas, são possíveis para variáveis reais. Neste método, a estrutura e parâmetros do modelo gráfico podem ser derivadas a partir do banco de dados. Critérios de avaliação do modelo são normalmente Bayesianos e a estimação de parâmetros pode ser uma mistura de estimativas fechadas e métodos iterativos dependendo se as variáveis são diretamente observadas ou escondidas. A busca no modelo consiste de algoritmos gulosos sobre diversas estruturas de grafos. Conhecimento anterior, como um ordenamento parcial entre as variáveis pode ser útil reduzindo o espaço de busca do modelo (GOEBEL, 1999).

Modelos de aprendizado relacional, também conhecidos como programação lógica indutiva, utilizam uma linguagem de padrões da lógica de primeira ordem mais flexível do que a lógica proposicional que é utlizada na representação de regras e árvores. O poder representacional extra dos modelos relacionais também exige maior demanda computacional para o seu processamento em termos de busca (FAYYAD, 1996).

Além dos métodos até aqui apresentados existem inúmeros outros que são particulares e especializados a determinados tipos de dados e domínios. Apesar de diversos algoritmos e aplicações parecerem diferentes superficialmente, não é raro que muitos deles dividam componentes em comum. Sendo assim, compreender mineração de dados e indução de modelos no nível apresentado torna claro o comportamento de qualquer algoritmo de mineração de dados e torna mais fácil entender sua contribuição geral e aplicabilidade ao processo de KDD (FAYYAD, 1996).

Um ponto importante é que cada técnica se encaixa melhor em determinados problemas do que em outros. Por exemplo, classificadores baseados em árvores de decisão podem ser úteis para encontrar estruturas em espaços de muitas dimensões e em problemas com dados contínuos e categóricos misturados, pois métodos baseados em

árvore não requerem medidas de distância. Contudo, este método pode não ser adequado para problemas onde os limites reais entre as classes são descritos por um polinômio de segunda ordem, por exemplo. Por este motivo, é importante que as diferentes técnicas sejam avaliadas a fim de se identificar qual a mais adequada para os objetivos finais. Se o objetivo é prever o comportamento de determinadas variáveis, por exemplo, a utilização de modelos que simplesmente descrevam os dados não será útil. Além disso, o tipo de aprendizado, a tarefa a ser realizada, o tipo de repositório a ser minerado e o formato de representação do conhecimento adquirido devem ser considerados durante a seleção da técnica de mineração de dados (GOEBEL, 1999).

Os dois principais **tipos de modelos** gerados na etapa da mineração de dados são o preditivo e o descritivo. O primeiro é gerado a partir de um conjunto de variáveis conhecidas e é utilizado para prever valores futuros de outras variáveis de interesse. O segundo, descreve os dados e deve ser interpretável pelo ser humano (CHEN, 2000).

Quanto ao **tipo de aprendizado**, os algoritmos podem ser classificados conforme a intervenção humana e conforme a manipulação dos dados de entrada. Segundo o primeiro critério, podem ser supervisionados ou não-supervisionados. No aprendizado supervisionado os dados são previamente classificados por um especialista no domínio da aplicação e são utilizados como exemplos pelo algoritmo de mineração de dados. O algoritmo analisa todos os exemplos de cada classe informada pelo especialista e procura um modelo que descreva esses dados de modo a classificar corretamente o maior número de casos possível. Já o aprendizado não-supervisionado busca identificar como os dados estão relacionados, quais itens são similares, quais são diferentes e de que forma. Eles agrupam padrões apresentados na entrada de acordo com a similaridade existente entre eles. Nesse caso, nenhuma classe é predefinida e cabe ao próprio algoritmo manipular os dados de entrada de modo a determiná-las. Segundo a forma de manipulação dos dados entrada, os algoritmos podem ser de aprendizado em lote ou incrementais. Algoritmos de aprendizado em lote consideram todo o conjunto de dados de uma única vez. A modificação dos dados de entrada implica na necessidade da aplicação do algoritmo novamente. Algoritmos baseados em aprendizado incremental não desconsideram resultados de minerações anteriores quando o arquivo de entrada é modificado (SILVA, 2003) (CHEN, 2000).

Quanto à **tarefa a ser realizada**, os algoritmos utilizados podem ser de classificação, regressão, agrupamento, análise de associações entre outros (GOEBEL, 1999).

Em tese, a mineração de dados pode ser aplicada a qualquer **tipo de repositório** de dados, mas os algoritmos e dificuldades de implementação variam bastante conforme este (HAN, 2001). Os principais tipos são bancos de dados relacionais, *data warehouses*, bancos de dados transacionais, bancos de dados orientados a objetos, bancos de dados geográficos, bancos de dados temporais, além de outros.

Por fim, sabe-se que os padrões identificados por técnicas de mineração de dados podem ser representados em diversos **formatos**, os quais são classificados em caixa transparente ou caixa preta. Formatos do tipo caixa transparente representam os padrões em termos de uma estrutura que pode ser examinada e entendida por humanos, sobre a qual ainda é possível raciocinar e pode ser usada para decisões futuras. Os do tipo caixa preta já não apresentam a estrutura do padrão de modo explícito. Os diferentes formatos podem variar entre tabelas de decisão, árvores de decisão, regras de classificação, regras associativas, centroide, e mais (WITTEN, 2005).

4 SELEÇÃO DA TÉCNICA DE DESCOBERTA DE CONHECIMENTO

Dada a introdução aos conceitos básicos de processo de KDD e a sua etapa de mineração de dados, este Capítulo apresenta as considerações realizadas na seleção da técnica de descoberta de conhecimento aplicada nas etapas subsequentes da presente pesquisa.

Inicialmente são apresentadas algumas considerações quanto à decisão de se aplicar um processo de KDD ao problema em questão. Também são feitas algumas observações sobre trabalhos similares e a abordagem proposta nesta pesquisa. Em seguida são apresentadas as análises realizadas para a escolha da classe de algoritmos e do algoritmo específico de mineração de dados a ser utilizado. Após a seleção do algoritmo, este é introduzido tendo as suas principais ideias e conceitos exibidos.

4.1 Uma Potencial Aplicação de KDD

Conforme Fayyad (1996), na seleção de uma potencial aplicação para um processo de KDD os critérios podem ser divididos em dois grupos: critérios práticos e critérios técnicos. Dentro dos critérios práticos estão aqueles mesmos encontrados em outras aplicações de tecnologia avançada, tais como o potencial impacto de uma aplicação, a inexistência de soluções alternativas mais simples, forte suporte organizacional para uso da tecnologia, entre outros. Para aplicações científicas o impacto pode ser medido através da novidade e qualidade do conhecimento descoberto, assim como através da melhoria no acesso a dados através da automação de um processo manual de análise. Para aplicações de negócios o impacto se traduz em aumento de receitas, menores custos e aumento de qualidade, por exemplo. Com relação ao suporte organizacional, este normalmente deve se dar através de um especialista no domínio que possa definir medidas próprias de interesse para aquele domínio e participar do processo de KDD. Aplicações que lidam com informações e dados pessoais ainda devem considerar questões legais e de privacidade.

Nos critérios técnicos estão considerações sobre a disponibilidade de dados (casos) suficientes. De modo geral, quanto mais campos e mais complexos os padrões procurados, mais dados são necessários. Conhecimento inicial é um fator que pode reduzir o número de casos necessários. Uma outra consideração a ser feita é sobre a relevância dos atributos. É importante que se tenham atributos relevantes para a tarefa de descoberta, caso contrário, não há massa de dados capaz de possibilitar predição a partir de atributos que não capturem a informação requerida. Além disso, baixos níveis de ruído, ou seja, poucos erros nos dados, são uma outra consideração. Grande quantidade de ruídos dificulta a identificação de padrões a menos que uma grande

quantidade de casos possa mitigar o erro aleatório e ajudar a clarear os padrões. Dados que mudam e dependentes do tempo, apesar de tornar o desenvolvimento da aplicação mais complexo, a tornam mais útil, pois é muito mais fácil retreinar um sistema do que um humano. Por fim, uma das considerações mais importantes é o conhecimento prévio disponível. É útil conhecer algo sobre o domínio, como quais os campos importantes, quais as relações esperadas, qual a função de utilidade para o usuário, quais padrões já são conhecidos e assim por diante.

Tendo-se estes critérios de avaliação por base, pode-se verificar que o impacto da aplicação de um processo de KDD para o objetivo proposto é bastante significativo. Este processo será capaz de trazer à tona um conhecimento novo do qual não se tem informações, isto é, dados a respeito das associações recorrentes entre os padrões de atividade em modelos de processos. Além disto, com estes resultados, será possível atingir o objetivo de automatizar parte da etapa manual de modelagem de processos de negócio através do auxílio ao desenhista com base nos padrões minerados.

Com relação à existência ou não de soluções mais simples, encontram-se outros trabalhos relacionados com mineração de processos de negócio. No entanto, a abordagem proposta aqui, de mineração de modelos, não é vista nestes trabalhos, já que a abordagem por eles utilizada baseia-se na mineração de *logs* de execução de processos (CASATI , 2004), (AALST, 2005b), (GÜNTHER, 2008), (TRISTÃO, 2008). Esta utilização de modelos no lugar de *logs* é uma vantagem bastante interessante na medida em que possibilita que informações sejam extraídas sem a execução dos processos, inclusive já durante a sua própria fase de modelagem.

A característica de suporte organizacional não faz muito sentido ser avaliada neste contexto. A aplicação do processo de KDD está se dando em um trabalho acadêmico e não em uma empresa. Contudo, pode-se considerar que o especialista no domínio são os pesquisadores, pois estes têm plenas condições para definir medidas de interesse e sempre participam de todo o processo, tendo todo o conhecimento necessário para a execução do mesmo.

Para tratar da privacidade dos dados utilizados é adotada a política de não se revelar as fontes dos processos de negócio utilizados, apenas o seu domínio de aplicação. Desta maneira, permite-se que os processos sejam estudados e relações entre suas características sejam feitas sem, no entanto, revelar a quem eles pertencem.

Entrando na análise dos critérios técnicos, verifica-se que já existe uma certa disponibilidade de dados, através de um grupo de variados modelos de processos. Quanto aos atributos a serem utilizados tem-se que aqueles relevantes são os padrões de atividade presentes nos modelos e suas conexões. Além disto, podem ser de interesse características de cada processo e domínios de aplicação destes.

Em Thom (2006b) tem-se um estudo utilizado como conhecimento inicial. Os padrões de atividade são apresentados, assim como relações causais binárias entre eles, por exemplo: em *x*% dos modelos analisados o padrão A é seguido pelo padrão B. Desta maneira já se sabe por quais tipos de relacionamento a busca deve começar, quais são alguns dos padrões de associação já conhecidos e o que se pode esperar encontrar com a aplicação do processo de KDD.

Para tratar dos baixos níveis de ruído tem-se que os processos analisados foram desenvolvidos por uma mesma organização, ou seja, utilizando-se os mesmos conceitos e padrões de modelagem. Uma desvantagem disto é que modelos desenhados por outros

grupos talvez tenham alguns vícios diferentes dos modelos estudados. Para contornar este fato de que as características de modelagem podem variar, a maior parte possível do trabalho será automatizada, o que permitirá que novos processos sejam rápida e facilmente submetidos à análise e capazes de gerar resultados.

Sendo assim, evidenciam-se grandes oportunidades de ganhos com a aplicação de um processo de KDD ao problema proposto, bem como se confirma o fato de que se possuem todas as ferramentas necessárias para a aplicação de tal processo. A automatização de boa parte deste também será de grande valia, na medida em que possibilitará a análise de diversos modelos de processo de negócio quando estes forem incluídos para a descoberta de novo conhecimento.

4.2 Classe de Algoritmos Selecionada

Tendo-se chegado à conclusão de que a aplicação de um processo de KDD é válida e pode trazer resultados interessantes passou-se a considerar qual a classe de algoritmos de mineração de dados deveria ser aplicada a fim de proporcionar os melhores resultados. Por melhores resultados, entende-se que são aqueles que atendem aos nossos requisitos iniciais e ofereçam suporte a todas as análises posteriores que se deseja fazer permitindo que os objetivos da pesquisa sejam atingidos.

Cada uma das variáveis dos algoritmos de mineração de dados apresentada na Subseção 3.1.2 foi considerada e chegou-se à classe de algoritmos que deveria ser aplicada para a obtenção do conhecimento desejado. A Tabela 4.1 apresenta as variáveis consideradas e os valores escolhidos. Com base nesta tabela é que o algoritmo de mineração de dados foi procurado e escolhido para ser aplicado.

Tabela 4.1: Variáveis dos algoritmos de mineração de dados

Variável	Valor
Tipo de modelo	Preditivo
Tipo de aprendizado	Não-supervisionado em lote
Tarefa a ser realizada	Análise de associações
Tipo de repositório a ser minerado	Bases de modelos de processos
Formato de representação do conhecimento	Caixa transparente

Como se pode notar pela primeira linha da Tabela 4.1, os algoritmos alvo de interesse para o presente trabalho são aqueles da classe dos algoritmos preditivos. Esta escolha foi realizada, pois em um dos objetivos o que se busca é uma previsão do futuro com base em fatos já ocorridos. Neste caso, com base em um modelo de *workflow* parcialmente projetado (fatos ocorridos), busca-se prever quais as próximas estruturas a serem acrescentadas ao modelo (previsão do futuro).

Como não se dispõe de um especialista de plantão, o ideal é que o algoritmo utilizado possua um aprendizado do tipo não-supervisionado de modo a reduzir ao máximo a intervenção humana. Além disso, como o conhecimento a ser minerado será baseado em uma base de dados definida e não necessitará ser realizado dinamicamente, o aprendizado pode ser em lote. Através desta abordagem uma série de processos pode ser preparada em um arquivo e este pode ser submetido à mineração. A utilização de um algoritmo incremental não se faz necessária, já que caso novos modelos estejam

disponíveis para a pesquisa, estes podem ser incluídos na base de dados e o algoritmo de mineração de dados pode ser rodado novamente.

Adicionalmente, é importante que os padrões derivados da fase de mineração, i.e., as co-ocorrências, sejam de fácil entendimento por humanos, o que permite que análises manuais também sejam executas. Assim, pode-se investigar relações entre as diversas co-ocorrências dos padrões de atividade e características específicas dos processos. Por exemplo, verificar se um processo manual, i.e., um processo onde a maioria das atividades são executadas por humanos, possui um conjunto de co-ocorrências de atividades diferente de um processo computadorizado, i.e., um processo onde a maioria das atividades são automatizadas. Como se tem um interesse futuro também em investigar diferentes classes de modelos de processo com base nas co-ocorrências de padrões de atividades mais recorrentes, é interessante que os algoritmos aplicados ao longo do processo de descoberta de conhecimento ofereçam esta possibilidade. Por este motivo, definiu-se que a tarefa a ser realizada fosse do tipo de análise de associações. Além de ter um caráter preditivo, esta tarefa pode ser vista como uma descrição dos dados atualmente disponíveis e inferências sobre características dos modelos podem ser realizadas.

Conforme especificado na Tabela 4.1, o tipo de repositório a ser minerado é constituído por uma base de modelos de processos. É importante salientar que esta base não necessariamente está implementada em um sistema gerenciador de banco de dados, mas pode muito bem consistir de simples arquivos contendo os modelos.

Para facilitar a inferência humana, o formato de representação do conhecimento selecionado foi do tipo caixa transparente. Neste modelo, o conhecimento é explicitamente representado, o que deixa a informação descoberta em uma maneira legível. Isto permite que mais dados sobre os modelos minerados possam ser descobertos quando um humano analisa o resultado.

Em virtude destas observações partiu-se para o estudo da classe de algoritmos de **descoberta de regras de associação**. Esta técnica é baseada em aprendizado não-supervisionado e dispensa, portanto, a classificação prévia dos dados de entrada e a interferência do especialista nesta atividade. Além disto, seus resultados são facilmente compreendidos pelo ser humano, seus resultados representam relações empíricas e é possível evidenciar o valor da mineração através de medidas especiais como suporte e confiança, por exemplo (AGRAWAL, 1993), (HAN, 2001).

Regras de associação são declarações da forma "*98% das pessoas que compraram pão e café, também compraram leite*". O algoritmo básico e representativo desta classe é o Apriori. Ele busca regras de associação entre itens de um banco de dados e uma de suas principais aplicações tem sido na análise de itens de uma cesta de compras, i,e., itens comprados por um consumidor (AGRAWAL, 1993).

Uma regra de associação é uma expressão da forma X => Y, onde X (antecedente) e Y (consequente) são conjuntos de itens. O problema da mineração de regras de associação pode ser visto como: dado um banco de dados D de transações, onde cada transação T é formada por um conjunto de itens, devem-se encontrar todas as regras de associação da forma X => Y que possuem um determinado suporte e confiança mínimos especificados pelo usuário. Uma regra X => Y possui uma confiança *c*, se *c*% das transações em D que contêm X, também contêm Y. A mesma regra possui um suporte *s*, se *s*% das transações em D contêm X U Y.

O problema de descobrir todas as regras com suporte e confiança mínimos pode ser dividido em dois subproblemas (AGRAWAL, 1993):

1. Encontrar todos os conjuntos de itens que possuem um suporte maior do que o suporte mínimo, ou seja, encontrar os **conjuntos de itens frequentes** (HAN, 2000). É por esta etapa que o algoritmo Apriori é responsável.

2. Utilizar os conjuntos de itens frequentes para gerar as regras desejadas. A ideia geral é que se ABCD e AB são conjuntos frequentes, então se pode determinar se a regra AB => CD é válida computando a razão *r* = suporte(ABCD)/suporte(AB). Se *r* for maior do que a confiança mínima, então a regra é válida. O suporte mínimo está garantido porque ABCD é um conjunto frequente.

O funcionamento do algoritmo Apriori pode ser entendido da seguinte maneira (AGRAWAL, 1993):

- Uma primeira passada do algoritmo conta o número de ocorrências de cada item para determinar os conjuntos frequentes de tamanho 1.
- Cada passada *k* subsequente considera somente os conjuntos frequentes encontrados na passada anterior. A propriedade combinatorial básica utilizada é que qualquer subconjunto de um conjunto frequente também precisa ser um conjunto frequente. Assim, os conjuntos frequentes de tamanho *k* podem ser gerados juntando conjuntos de *k*-1 itens, e removendo aqueles que contêm qualquer subconjunto que não seja frequente.

Entretanto, o algoritmo Apriori básico foi desenvolvido para ser aplicado sobre bancos de dados transacionais, isto é, bancos de dados compostos por transações, como o caso das cestas de compras. Modelos de processos de negócio não são transações e, portanto, o algoritmo Apriori básico não pode ser aplicado neste caso.

Dentre as possíveis alternativas para se minerar regras associativas a partir de modelos de processos, uma delas é mapear um modelo para uma cesta de compras, onde cada estrutura do modelo representaria um item da cesta. Por exemplo, o simples modelo da Figura 4.1 seria mapeado para uma cesta de compras com os seguintes itens: 1 nó de início, 1 padrão notificação, 1 padrão bi-direcional, 1 padrão de notificação, 1 padrão unidirecional e um nó de fim.

Figura 4.1: Modelo exemplo a ser mapeado para cesta de compras

Contudo, uma falha neste mapeamento pode logo ser detectada: ele não captura as conexões que existem entre os parâmetros. Aprimorando-se este mapeamento, pode-se ter a seguinte cesta de compras: 1 nó de início na posição 1, 1 padrão de notificação na posição 2, 1 conexão entre as posições 1 e 2 e assim sucessivamente. Novamente, este mapeamento apresenta uma falha: padrões idênticos, mas que ocorrem em posições

diferentes dos modelos de *workflow* não terão a sua correspondência identificada na utilização de um simples algoritmo como o Apriori. Em outras palavras, utilizando-se esta abordagem o Apriori não é capaz de detectar que uma conexão entre um padrão de notificação na posição 2 e um padrão bi-direcional na posição 3 tem o mesmo significado que uma conexão entre um padrão de notificação na posição 5 e um padrão bi-direcional na posição 6.

Com este esforço, verifica-se que o *framework* baseado em transações do Apriori não pode ser utilizado para minerar de maneira efetiva conjuntos de dados de domínios diferentes. Em vista disto, evidencia-se a necessidade da utilização de algoritmos modificados e voltados a outros domínios, que não a cesta de compras.

Olhando-se para a ideia básica do Apriori, pode-se observar que o conjunto de itens frequentes pode ser uma abstração para estruturas mais complexas. Em outras palavras, os elementos dos conjuntos de itens frequentes não necessariamente precisam ser simples itens de uma cesta de compras, mas podem ser elementos com algum significado ou estrutura em algum contexto. Pesquisando-se nesta linha de raciocínio chega-se a diversos algoritmos para mineração de itens frequentes em diferentes domínios, tais como: redes sociais, dados sequenciais, dados temporais, cadeias de proteína, compostos químicos e, os de maior interesse para o presente trabalho, **grafos** (PADMANABHAN, 2005), (JIMÉNEZ, 2007). De fato, ao longo dos últimos anos, algoritmos de descoberta de itens frequentes têm sido aplicados para encontrar padrões interessantes em várias áreas de aplicação (KURAMOCHI, 2004).

Com uma visão mais teórica pode-se notar que um modelo de processo de negócio nada mais é do que um grafo com uma semântica pré-definida. Dentro deste modelo, um caminho para se formular o problema da descoberta de padrões frequentes é o problema de descobrir subgrafos que ocorrem frequentemente no conjunto de grafos. Algoritmos de mineração de grafos fazem exatamente isto. Dado um conjunto de grafos, estes algoritmos procuram por subestruturas que satisfaçam alguns critérios, tais como frequência mínima ou confiança mínima (KURAMOCHI, 2004).

A mineração de grafos ainda é vantajosa, pois grafos são capazes de representar relações complexas. Devido a sua característica, ela tem a possibilidade de utilizar a estrutura natural do domínio de aplicação e minerar diretamente sobre esta estrutura (PADMANABHAN, 2005), não necessitando de mapeamentos como outros algoritmos podem exigir e conforme foi verificado ao tentar-se aplicar o Apriori na mineração de modelos de processos.

4.3 Algoritmo selecionado

Existe uma grande e crescente gama de algoritmos para a mineração de conjuntos frequentes em grafos. Todos se baseiam no mesmo princípio fundamental e chegam ao mesmo resultado. De fato, todos eles chegam ao mesmo resultado, pois a definição de conjunto frequente é clara e não ambígua. Desta forma, se um algoritmo não apresentar todos os conjuntos frequentes como saída, ele não é um algoritmo completo (CHAKRAVARTHY, 2004), (FISCHER, 2004), (KURAMOCHI, 2004), (CHAKRAVARTHY, 2008).

As diferentes variações destes algoritmos devem-se, sobretudo, à incessante busca por melhores desempenhos em massas de dados cada vez mais densas. A aplicação/adaptação de algoritmos como o FP-Growth (HAN, 2000), suprimindo a etapa

explícita da geração de candidatos, ainda é um problema em aberto no ramo de mineração de grafos (KURAMOCHI, 2004). Outro ponto que diferencia os algoritmos desenvolvidos para este propósito é onde o algoritmo é aplicado: existem aqueles desenvolvidos em programas de aplicação que acessam um banco de dados e aqueles desenvolvidos para rodar diretamente sobre o banco de dados (PADMANABHAN, 2005), (CHAKRAVARTHY, 2008).

Com base nisto, o algoritmo selecionado foi o FSG (*Frequent SubGraph*) (KURAMOCHI, 2004). Este é um algoritmo computacionalmente eficiente desenvolvido para a descoberta de todos os subgrafos frequentes em grandes bases de dados de grafos não dirigidos. Uma observação importante com relação à escolha das características do algoritmo selecionado é o fato de se ter optado por um algoritmo de mineração de grafos não dirigidos. Esta decisão vem em decorrência de um dos objetivos a ser atingido com a utilização dos subgrafos frequentes: prever evoluções de modelos de processos parciais. Para prever esta avolução, o que é buscado, em última instância, é encontrar um grafo maior (modelo de processo evoluído) que contenha um grafo menor (modelo de processo parcial). A utilização de grafos não direcionados permite que ocorrências como as da Figura 4.2 sejam consideradas iguais. Isto permite que associações entre padrões com a mesma topologia sejam agrupadas, abstraindo-se os sentidos das arestas, o que leva a um suporte mais alto para os subgrafos frequentes. Com isto, também são gerados menos padrões frequentes, tornando a fase de análise mais fácil de ser realizada e permitindo uma execução mais rápida de um algoritmo de sugestões automatizadas que deve varrer todo o conhecimento minerado.

Figura 4.2: Grafos que são considerados iguais quando a direção não é levada em conta

Com esta abordagem, a partir dos padrões minerados será possível especificar qual o mais provável grafo que engloba um grafo parcial. Considere-se que a Figura 4.3 (a) representa o que seria um modelo de processo desejado pelo usuário e o item (b) representa um processo parcialmente por ele modelado. Os itens (c) e (d) representam padrões frequentes encontrados pelo FSG que são possíveis evoluções do processo representado em (b) para um modelo final desejado, na medida em que representam grafos que são evoluções daquele em (b), abstraindo-se o sentido das arestas. Ou seja, pode-se ver o modelo parcial, apresentado em (b), como contido nos grafos (c) e (d). Utilizando-se estes padrões frequentes, pode-se apresentá-los como sugestões de evolução do modelo parcialmente desenhado ao usuário, deixando a seu critério a definição do sentido das arestas, assim como a definição de outros atributos específicos de seu processo particular, tais como participantes e descrições específicas de atividades.

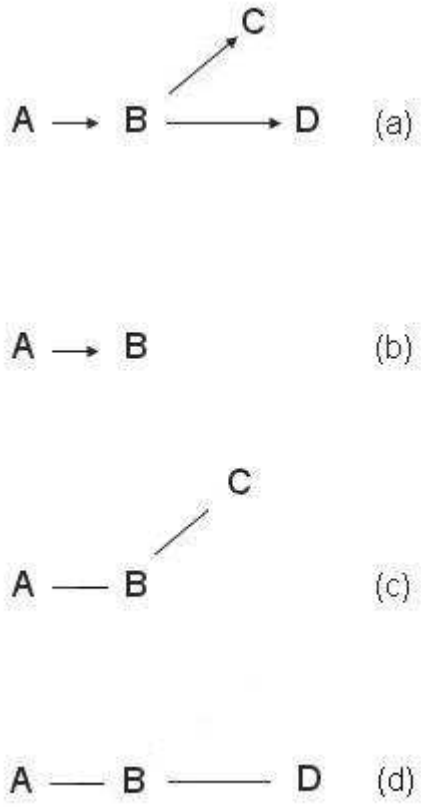

Figura 4.3: Sugestão de evolução

Além das considerações e expectativas apresentadas acima, pode-se afirmar que algoritmos para a mineração de subgrafos dirigidos são mais complexos, exigem um maior tempo de processamento (KURAMOCHI, 2004) e são mais difíceis de serem encontrados. Em virtude do exposto até aqui e do fato de que o desenvolvimento de um novo algoritmo de mineração de grafos não é o objetivo deste trabalho, optou-se por utilizar um algoritmo para mineração de grafos não dirigidos.

O FSG é interessante, pois apresenta algumas otimizações inteligentes sobre a ideia básica dos algoritmos de mineração de grafos e utiliza-as para obter um melhor desempenho, tais como: a) utilização de uma representação de grafos esparsos que minimiza o espaço e computações requeridos; b) ele aumenta o tamanho dos grafos frequentes adicionando uma aresta por vez, o que permite a geração eficiente de candidatos; c) ele incorpora várias otimizações na geração de candidatos e contagem de frequência possibilitando sua escalabilidade para grandes bancos de dados sobre grafos, e; d) ele utiliza algoritmos sofisticados de etiquetação canônica, tais como invariantes de vértices e ordenamento de particionamento baseado em grau para identificar unicamente os vários subgrafos gerados sem ter de refazer procedimentos caros de computação de isomorfismo de grafos e subgrafos. Algoritmos mais eficientes geralmente são mais complexos de serem implementados e envolvem uma carga de teorias relacionadas maior do que algoritmos mais simples. Além disso, determinados algoritmos são mais eficientes somente para um conjunto de dados de entrada com características específicas. Como não se tem o conhecimento do desenvolvimento de um algoritmo específico para a mineração de padrões frequentes em modelos de processos,

o algoritmo FSG foi escolhido por ter uma boa documentação, entrada e saída simples e bem definidas e fácil acesso a sua implementação.

4.3.1 O FSG (KURAMOCHI, 2004)

Dada uma base de grafos, um algoritmo de mineração de grafos procura por subestruturas que satisfaçam alguns critérios, tais como frequência mínima, confiança mínima, interesse mínimo e frequência máxima (NIJSSEN, 2004). Conforme Karypis (2004), um grafo G = (V, E) é composto por dois conjuntos: o conjunto de vértices *V* e o conjunto de arestas *E*. Cada aresta é um par de vértices e, nos grafos não dirigidos, este par é não ordenado. Um grafo ainda pode ser etiquetado, isto é, cada vértice e aresta pode possuir associado a ele uma etiqueta a partir de um conjunto de etiquetas de vértices (*Lv*) e etiquetas de arestas (*Le*). Não é requerido que cada vértice, ou aresta, possua uma etiqueta única e a mesma etiqueta pode ser associada a múltiplos vértices, ou arestas, no mesmo grafo.

Dado um grafo G = (V, E), um grafo Gs = (Vs, Es) será um subgrafo de G se, e somente se, Vs estiver contido em V e Es estiver contido em E. Um grafo é conexo se existe um caminho entre cada par de vértices no grafo.

Adicionalmente, dois grafos G1 = (V1, E1) e G2 = (V2, E2) são isomorfos se eles são topologicamente idênticos, isto é, se existe um mapeamento de V1 para V2 tal que cada aresta de E1 é mapeada para uma única aresta de E2 e vice-versa. No caso dos grafos etiquetados este mapeamento também deve preservar as etiquetas dos vértices e arestas. Um automorfismo é um mapeamento de isomorfismo onde G1 = G2. Dados dois grafos G1 = (V1, E1) e G2 = (V2, E2) o problema de isomorfismo de subgrafos é encontrar um isomorfismo entre G2 e um subgrafo de G1, isto é, determinar quando G2 está incluído em G1 ou não.

A etiquetação canônica de um grafo G = (V, E), *cl(G)*, é definida como um código único (uma *string*) que é invariante com relação ao ordenamento dos vértices e arestas do grafo. Como resultado, dois grafos terão a mesma etiqueta canônica se eles forem isomorfos. Etiquetas canônicas são de extrema importância na medida em que possibilitam rapidamente comparar dois grafos e estabelecer uma ordenação completa de um conjunto de grafos de modo único e determinístico, independentemente da ordenação inicial dos vértices e arestas. Uma vez obtida a etiquetação canônica de um grafo este pode ser facilmente comparado com outros sem a necessidade de recalculá-la.

Uma maneira simples de se gerar um código para um grafo é converter a sua matriz de adjacência em uma sequência linear de símbolos. Por exemplo, isto pode ser obtido concatenando-se as linhas ou colunas da matriz de adjacência de um grafo para obter-se uma sequência de etiquetas de vértices e arestas. A Figura 4.4 ilustra exemplos para um grafo não etiquetado G_6 (a) e um grafo etiquetado que possui etiquetas de vértices e arestas G_3 (c). O símbolo v_i representa o identificador de um nodo e elementos em branco na matriz de adjacência indicam que não existem arestas entre o par de vértices correspondente. A matriz de adjacência do grafo G_6 (b) produz um código “000000101011000100010” através da concatenação da lista das seis etiquetas dos vértices “000000” e as colunas do triângulo superior da matriz de adjacência “1”, “01”, “011”, “0001” e “00010” onde cada vértice é representado por “0” e cada aresta por “1”. A inexistência de arestas é representada por “0”. De forma semelhante, a matriz de adjacência (d) produz o código “aaazxy”. As três primeiras letras “aaa” são as etiquetas

dos vértices na ordem em que aparecem na matriz de adjacência. As outras três letras são obtidas a partir da concatenação das colunas do triângulo superior.

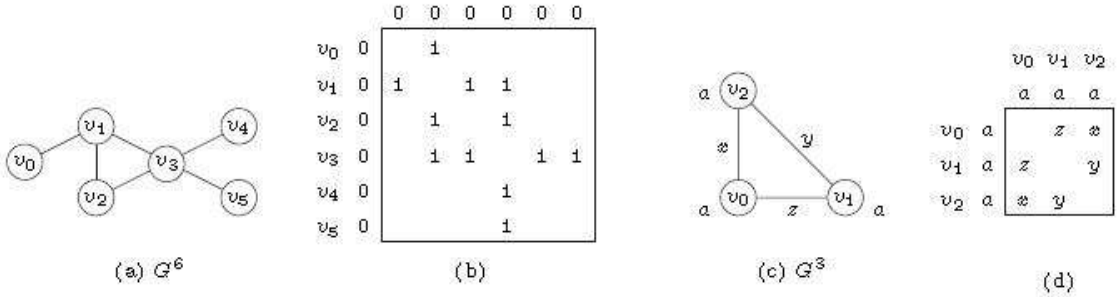

Figura 4.4: Exemplos de geração de códigos para grafos (KURAMOCHI, 2004)

Um fato extremamente importante a ser notado é que os códigos gerados na Figura 4.4 não podem ser utilizados como etiquetas canônicas, porque eles são dependentes da ordenação dos vértices. Permutações das linhas e colunas da matriz de adjacência dariam origem a diferentes códigos, o que viola o requisito das etiquetas canônicas que devem ser invariantes com relação a isomorfismos. Uma maneira para obter-se códigos invariantes com relação a isomorfismo é gerar todos os códigos possíveis através das permutações dos vértices e escolher o menor, ou o maior, código lexicográfico (FORTIN apud KURAMOCHI, 2004).

A ordenação dos vértices dos grafos da Figura 4.4 que retorna o maior código é exibida na Figura 4.5. Os códigos das matrizes de adjacência são “00000011110010000100 0” para a matriz (a) e “aaazyx” para a matriz (b). Estes códigos podem ser utilizados como etiquetas canônicas dos grafos, pois sempre que se procurar pelo maior código destes grafos, chegar-se-á ao mesmo resultado.

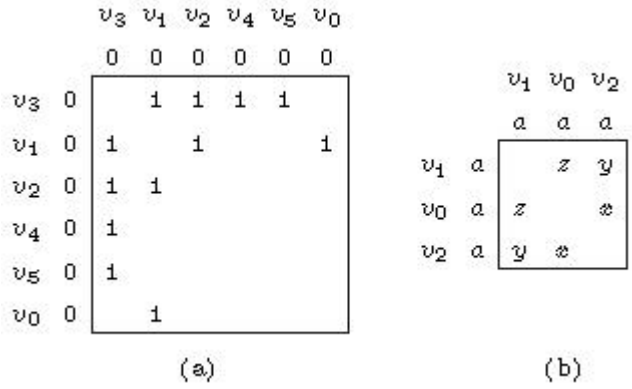

Figura 4.5: Matrizes canônicas de adjacência

Computar a etiqueta canônica conforme o método apresentado possui uma complexidade de *n!* para um grafo com *n* nodos, já que todas as *n!* permutações dos nodos necessitam serem averiguadas antes de selecionar-se o maior código. Para reduzir a complexidade deste algoritmo o FSG utiliza a técnica dos invariantes de vértices.

Invariantes de vértices são alguns atributos ou propriedades de um vértice que não mudam entre mapeamentos isomórficos. Um exemplo de uma propriedade invariante com relação a isomorfismos é o grau ou a etiqueta de um vértice, os quais permanecem os mesmos independentemente do mapeamento adotado.

Os invariantes de vértices podem ser utilizados para reduzir o tempo necessário para computar uma etiqueta canônica determinando partições dos vértices do grafo em classes de equivalência, tais que todos os vértices pertencentes a uma mesma partição possuam os mesmos valores para as propriedades invariantes de vértice consideradas. Como estas propriedades permanecem sempre as mesmas independentemente do ordenamento dos vértices, as mesmas partições podem ser criadas a partir de qualquer ordem. Utilizando-se estas partições pode-se definir a etiqueta canônica de um grafo como sendo o maior código lexicográfico obtido através da concatenação das colunas do triângulo superior das matrizes de adjacência, sobre todas as permutações de vértices sujeito a restrição de que os vértices de cada uma das partições devem ser numerados consecutivamente. Portanto, a única modificação sobre a definição anterior é que em vez de maximizar o código sobre todas as permutações de vértices, deve-se maximizar o código somente sobre as permutações que mantêm os vértices de uma mesma partição juntos.

Se *m* é o número de partições criadas utilizando os invariantes de vértices, contendo *p1, p2, ..., pn* vértices cada uma, então o número de permutações diferentes que devem ser consideradas é dado pelo produtório de *pi!*, com *i* variando de 1 até *m*, o que pode ser consideravelmente menor do que *n!* conforme requerido pelo método anterior. É fácil notar que os invariantes de vértices não reduzem a complexidade computacional assintótica do método, a qual permanece sendo da ordem de fatorial. Contudo, conforme Kuramochi (2004), para a maioria dos grafos do mundo real a quantidade de tempo requerida para o cálculo das etiquetas canônicas pode ser substancialmente reduzida, dado que se pode descobrir invariantes de vértices que levam a um grande número de patições.

Um invariante de vértice bastante simples e interessante é a etiqueta do vértice e o seu grau. Através destes atributos pode-se gerar uma partição para cada grau e etiqueta e ordenar as partições com base na ordem dos graus e na ordem lexicográfica das suas etiquetas. A Figura 4.6 apresenta um grafo com quatro vértices (a), sua matriz de adjacência ordenada conforme a ordem lexicográfica dos identificadores dos vértices (b), as partições destes vértices considerando-se os seus graus e suas etiquetas (c) e a matriz canônica de adjacência (d). Na Figura 4.6 (c) pode-se notar que os vértices estão particionados em três grupos: p0 = {v1}, p1 = {v0, v3} e p2 = {v2}. A partição p0 contém um vértice de grau três com a etiqueta *a*, p1 contém dois vértices de grau um cuja etiqueta é *a* e p2 contém um vértice de grau um com a etiqueta *b*. A matriz canônica de adjacência é obtida permutando-se os vértices dada a restrição de que a ordem das partições deve ser mantida. Desta forma, devem ser testadas 1!.2!.1! = 2 permutações, enquanto que o número total de permutações possíveis é 4! = 24.

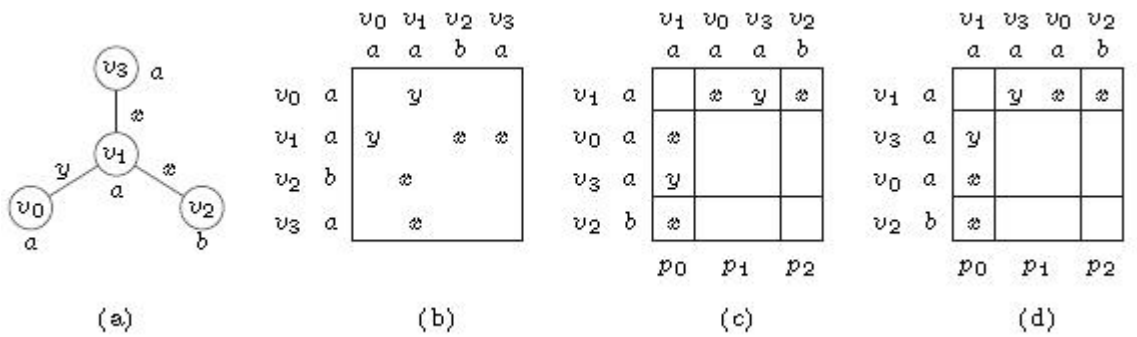

Figura 4.6: Um grafo e suas matrizes de adjacência

O problema de se determinar a etiquetação canônica de um grafo é equivalente a determinar o isomorfismo entre grafos, pois se dois grafos são isomorfos um ao outro, então as suas etiquetações canônicas devem retornar o mesmo resultado. Sobre estes problemas não se sabe dizer se são do tipo P ou NP-completos. Para resolver-se o problema de encontrar subgrafos frequentes deve-se resolver o problema de identificar corretamente como um subgrafo particular mapeia para os vértices e arestas de outro grafo. Este problema só pode ser solucionado resolvendo-se múltiplas instâncias do problema de isomorfismo de subgrafos, o que se tem mostrado estar na categoria dos problemas NP-completos (KURAMOCHI, 2004).

5 APLICAÇÃO DO PROCESSO DE DESCOBERTA DE CONHECIMENTO

Este Capítulo apresentará as etapas iniciais da aplicação do processo de descoberta de conhecimento sobre a base de modelos de processos de negócio disponível. Serão abordadas as etapas de seleção, pré-processamento, transformação e mineração dos dados.

Um passo importante que é apresentado neste Capítulo é a mineração dos modelos de processos. Esta abordagem requer algumas manipulações dos dados para que os resultados sejam obtidos e este trabalho será aqui apresentado. A utilização de modelos é um diferencial desta pesquisa em relação a outras abordagens conforme comentado no Capítulo 4, ao permitir que informações sejam extraídas tão logo os processos já tenham um modelo. Outras alternativas necessitariam ainda executar estes processos para obter *logs* de execução e então minerar padrões a partir destas informações.

Após a mineração, algumas observações já serão feitas a respeito de resultados do processo. A avaliação completa dos padrões encontrados será realizada no Capítulo 6.

5.1 Seleção, Pré-processamento e Transformação de Dados

A aplicação de um processo de descoberta de conhecimento tem o seu início com a preparação dos dados (FAYYAD, 1996). É nesta fase que os dados são selecionados, pré-processados e transformados.

Para a seleção dos dados que comporiam a base de modelos de processos a ser minerada realizou-se uma análise sobre os modelos disponíveis para o estudo do grupo de pesquisa e novas opções foram procuradas. A seleção foi bem abrangente e procurou incluir todos os modelos de processos possíveis. A partir disto foram obtidos 190 processos de negócio. Estes processos foram modelados no Oracle Builder e são executados em 12 organizações relacionadas a diferentes domínios de aplicação.

Em mais detalhes, 11 destes processos são executados em uma companhia grande e pouco centralizada. Eles referem-se ao gerenciamento do programa de qualidade total e outras atividades desta empresa. Outros 17 estão relacionados com o gerenciamento de atividades internas de uma fábrica de *software*, tais como, edição de *newsletter*, *feedback* de colaboradores e solicitação de férias.

Um conjunto de 133 são executados em 6 companhias grandes e altamente centralizadas. Destes, 33 referem-se ao processo de licenciamento ambiental de uma

organização governamental, 63 referem-se ao gerenciamento de atividades internas, como *help desk* e aprovação de solicitação de viagens, 32 referem-se ao gerenciamento de documentos, tais como escrita de cartas e relatório de reuniões, e 5 deles referem-se ao controle de direitos de acesso a *software* em uma fumageira.

Por razões de confidencialidade, 29 dos outros processos não puderam ser informados em que tipo de organização são executados. Estes se referem, sobretudo, a atividades de *help desk*, gerenciamento de *feedback* de usuários e aprovação de documentos. A Tabela 5.1 resume as características dos processos minerados.

Tabela 5.1: Características dos processos minerados

Tamanho das companhias	Tipo de tomada de decisão	Domínio de aplicação	Quantidade de modelos de processos
1 grande	Descentralizada	Gerenciamento de Qualidade Total e de atividades	11
1 pequena	Descentralizada	Gerenciamento de atividades internas	17
6 grandes	Centralizada	Gerenciamento de Qualidade Total e documentos e controle de acesso a software.	133
4 grandes	Indisponível	*Help Desk*, *feedback* de usuários e aprovação de documentos	29

Na etapa de pré-processamento, os padrões de atividade foram identificados manualmente com círculos e legendas em todos os processos analisados. Cada atividade, ou ordem parcial de atividades, foi identificada como um padrão de atividade conforme o contexto. A Figura 5.1 apresenta um exemplo desta identificação.

Figura 5.1: Identificação dos padrões de atividade nos modelos de processos

É importante observar que a mineração será realizada diretamente sobre os modelos dos processos, e não através de instâncias ou *logs* de execução como tem acontecido em alguns trabalhos e ferramentas. Softwares como o ProM ou o MinAdept apenas analisam resultados a partir das execuções dos processos e não extraem informações relacionadas à semântica e lógica (interna) (AALST, 2003), (CASATI, 2004), (ELLIS, 2006), (GÜNTHER, 2008), (LI, 2008), (TRISTÃO, 2008). Para que este novo tipo de mineração de processos fosse possível, os modelos de processos de negócio tiveram de

ser transformados para uma notação com a qual fosse possível trabalhar na extração dos padrões de forma mais facilitada. Nesta fase, apenas as informações de interesse de cada modelo de processo foram extraídas e mapeadas para uma linguagem capaz de representar grafos.

Na transformação dos modelos para grafos foram traduzidos quatro tipos diferentes de arestas e dezesseis tipos diferentes de nodos. Dentre os quatro tipos de arestas necessários para representar todos os modelos de processos selecionados para o estudo estão:

1. Normal (N): aresta normal entre dois nodos.
2. Bidirecional (B): representa duas arestas, uma em cada sentido entre os dois nós.
3. Dupla (D): representa duas arestas entre dois nodos.
4. Tripla (T): representa três arestas entre dois nodos.

Nos dezesseis tipos diferentes de nodos que aparecem nos modelos de processo selecionados encontram-se:

1. Start (S): nó representando o início do processo.
2. Aprovação (A): nó representando o padrão de aprovação.
3. Bidirecional (B): nó representando o padrão performativo bidirecional.
4. Decisão (D): nó representando o padrão de tomada de decisão.
5. Informativo (I): nó representando o padrão informativo.
6. Notificação (N): nó representando o padrão de notificação.
7. Unidirecional (U): nó representando o padrão performativo unidirecional.
8. Q&A (Q): nó representando o padrão de pergunta e resposta.
9. Financeiro (F): nó representando o padrão financeiro.
10. End 0 (E0): nó representando o encerramento normal de um processo.
11. End 1 (E1): nó representando o encerramento de um processo com uma semântica de aprovação.
12. End 2 (E2): nó representando o encerramento de um processo com uma semântica de cancelamento.
13. End 3 (E3): nó representando o encerramento de um processo com uma semântica de reprovação.
14. And Join (AJ): nó representando um conector do tipo AND-JOIN.
15. Or Join (OJ): nó representando um conector do tipo OR-JOIN.
16. Wait (W): nó representando uma parada no processo.

Na etapa de transformação do processo de KDD, os modelos de processos com os padrões de atividade identificados foram então já mapeados para o formato do arquivo de entrada do algoritmo escolhido para fazer a mineração de dados. Este é um arquivo no formato ASCII e deve conter todas as transações a partir das quais os padrões frequentes serão procurados. Cada transação representa um grafo e começa com a linha de transação, a qual é indicada através da letra *t*. Em seguida, aparece um conjunto de

linhas de vértices, as quais são iniciadas por *v* e possuem um índice do vértice e a sua etiqueta. Por fim, um conjunto de linhas de arestas, as quais iniciam por *u* e possuem os índices dos dois vértices da aresta e a sua etiqueta. Comentários podem ser inseridos neste arquivo com a utilização do caractere # na linha que se deseja comentar. Índices de vértice são inteiros não negativos, associados sequencialmente iniciando a partir do zero. Se um grafo possui *N* vértices, então devem existir *N* linhas de vértices cujos índices vão de 0 até *N*-1. Se um grafo contém *M* arestas, então devem existir *M* linhas de arestas, cada uma contendo um par de índices de vértices.

O modelo da Figura 5.1 seria mapeado para o formato apresentado na Figura 5.2:

```
   t # Exemplo – 1 - Processo de aprovação de campanha de
marketing de novo produto
   v 0 S
   v 1 N
   v 2 B
   v 3 N
   v 4 U
   v 5 E0
   u 0 1 N
   u 1 2 N
   u 2 3 N
   u 3 4 N
```

Figura 5.2: Mapeamento do processo de aprovação de campanha de marketing de novo produto

Neste exemplo, tem-se o mapeamento de um modelo completo. Primeiramente os nós são mapeados e cada um recebe um identificador inteiro incremental e uma etiqueta. As etiquetas são referências ao tipo de estrutura que um nó representa: S (nó de início), N (padrão de notificação), B (padrão bi-direcional), U (padrão unidirecional) e E0 (nó de fim). As arestas possuem o índice dos nós que cada uma delas conecta e todas possuem a mesma etiqueta *N*. Além disto, convém salientar-se que o comentário após o *t* da linha de transação segue um padrão para facilitar análises futuras: [Origem do processo] – [Identificador único] – [Descrição do Processo]. Esta padronização permite que aplicações automatizadas possam utilizar estes dados e obter maiores informações sobre os modelos de processos, possibilitando, inclusive, a realização de diferentes estudos.

Seguindo este mesmo modelo da Figura 5.2, todos os modelos de processos disponíveis foram mapeados para então informações serem extraídas e o conhecimento esperado minerado. O conjunto de todos os modelos de processos transformados para esta notação constitui a base de dados sobre a qual o processo de mineração de dados será executado e padrões serão obtidos.

5.2 Mineração de Dados

Após ter os dados preparados a etapa de mineração de dados pode ser iniciada. Como já apresentado no Capítulo 4, o algoritmo responsável por fornecer parte dos

resultados nesta etapa será o FSG. Partindo-se da base de dados construída conforme a transformação apresentada na Seção 5.1, o FSG foi executado e seus resultados foram então extraídos. A Figura 5.3 ilustra a saída da execução do algoritmo para a base de modelos disponível.

```
*************************************************
fsg 1.37 (PAFI 1.0) Copyright 2003, Regents of the University of Minnesota

Transaction File Information ------------------------
  Number of Distinct Edge Labels:              4
  Number of Distinct Vertex Labels:            16
  Average Number of Vertices In a Transaction: 11
  Max Number of Edges In a Transaction:        48
  Max Number of Vertices In a Transaction:     43

Options ---------------------------------------------
  Min Output Pattern Size:                     1
  Max Output Pattern Size:                     2147483647(INT_MAX)
  Min Support Threshold:                       2.0% (3 transactions)
  Generate Only Maximal Patterns:              No
  Generate PC-List:                            Yes
  Generate TID-List:                           Yes
Outputs ---------------------------------------------
  Frequent Pattern File:                       base.fp
  PC-List File:                                base.pc
  TID-List File:                               base.tid

  Size Candidates Frequent Patterns
  1              66
  2              224
  3    1719      331
  4    997       260
  5    467       377
  6    712       678
  7    1109      1096
  8    1450      1439
  9    1482      1480
  10   1164      1164
  11   680       680
  12   286       286
  13   82        82
  14   14        14
  15   1         1

  Largest Frequent Pattern Size:               15
  Total Number of Candidates Generated:        10163
  Total Number of Frequent Patterns Found:     8178

Timing Information ----------------------------------
  Elapsed User CPU Time:                       2.9[sec]
*************************************************
```

Figura 5.3: Execução do FSG

Como se pode ver na seção de informações sobre o arquivo de transações (*Transaction File Information*), dentro dos modelos submetidos à mineração de dados encontramos os quatro diferentes tipos de arestas e dezesseis tipos diferentes de nós que

foram apresentados durante a etapa de transformação na Seção 5.1. Nesta saída, o FSG já apresenta algumas informações estatísticas interessantes sobre a base, tais como o número médio de nós (onze, no caso estudado). Além disso, é informado que o processo com mais arestas possui 48 delas, enquanto que o processo com o maior número de vértices possui 43 deles.

Na seção de opções (*Options*) pode-se ver que os tamanhos mínimos e máximos dos padrões a serem encontrados foram deixados em seus valores *default* durante as execuções do FSG, pois não se deseja restringir as descobertas apenas a certos tamanhos de padrões. O ideal é que o algoritmo seja utilizado em seu maior potencial e, posteriormente, as informações extraídas sejam analisadas. Desta forma, o menor padrão considerado é aquele de tamanho um, ou seja, com uma aresta, enquanto que o maior padrão possível de ser encontrado é aquele cujo tamanho é igual ao máximo inteiro representável em 32 bits, 2.147.483.647

Pode-se observar que o suporte mínimo especificado para esta execução foi de 2%, o que, truncando-se, representa três transações na base utilizada. Este valor foi escolhido baixo para que um grande número de subgrafos frequentes fosse gerado e estes pudessem ser analisados. Os valores de suporte mínimo zero e um fariam com que todos os subgrafos possíveis fossem gerados, o que não é de interesse aqui, já que se buscam padrões frequentes. O suporte mínimo igual a dois requer apenas dois grafos com um subgrafo comum para que este seja exibido na saída, o que é um limite inferior demasiadamente baixo, considerando o universo de modelos a ser analisado, e que pode levar a uma massa de dados retornada muito volumosa para análises posteriores. Para efeitos comparativos, a base de dados em questão apresenta mais de 220.000 padrões frequentes se considerado um suporte mínimo de duas transações. Aumentando-se este suporte mínimo para três, o número de padrões frequentes retornados como saída cai para pouco mais de 8.000. Além de apresentar uma quantidade bem menor de padrões a serem analisados posteriormente, este resultado é obtido mais rapidamente: o cálculo de todos os padrões frequentes levou menos de três segundos para o suporte mínimo de três transações, enquanto que para um suporte mínimo igual a dois a operação leva mais de 30 minutos.

A Figura 5.4 ilustra parte do arquivo de saída do FSG que contém os padrões encontrados. Pode-se notar que as informações presentes neste arquivo são bastante semelhantes àquelas do arquivo de entrada, pois os resultados são nada mais que um conjunto de subgrafos, os quais também podem ser vistos como grafos.

Na Figura 5.4 é possível observar 4 padrões de tamanho 1 e um padrão de tamanho 2 com seus respectivos identificadores e suas frequências. Por exemplo, o primeiro padrão representado possui o identificador 1-0. Ele aparece em 48 modelos minerados. Este padrão minerado representa uma conexão entre um padrão bidirecional e um padrão unidirecional.

```
t # 1-0, 48
v 0 B
v 1 U
u 0 1 N
t # 1-1, 32
v 0 U
v 1 U
u 0 1 N
t # 1-2, 13
v 0 A
v 1 D
u 0 1 N
```

```
t # 1-3, 10
v 0 A
v 1 E3
u 0 1 N
t # 2-0, 18
v 0 U
v 1 B
v 2 U
u 0 1 N
u 0 2 N
```

Figura 5.4: Padrões encontrados

Seguindo na análise das informações retornadas pelo FSG na Figura 5.3, foi solicitado que este gerasse os arquivos de pais-e-filhos (*PC-List File*) dos padrões bem como o arquivo que indica quais transações contêm um determinado padrão (*TID-List File*). O primeiro é de grande importância para que a geração das regras de associação a partir de todos os conjuntos frequentes seja direcionada, evitando a geração de regras que não são de interesse. Para atingir este objetivo, o arquivo que contém as informações de quais padrões estão contidos em um determinado padrão maior, isto é, o *PC-List File*, deve ser utilizado para gerar apenas as regras que possuem o padrão menor (filho) como antecedente e o padrão maior (pai) como consequente. Desta maneira estarão sendo geradas regras de um passo apenas, isto é, que partem do grafo antecedente e adicionam apenas uma aresta, ou uma aresta e um nodo, para gerar o grafo consequente. A Figura 5.5 apresenta uma linha do arquivo de pais-e-filhos dos padrões. Esta linha diz que o padrão 2-0 é pai dos padrões 1-0 e 1-1, ou seja, os padrões 1-0 e 1-1 estão contidos no padrão 2-0. Logo, duas regras podem ser derivadas a partir desta informação: uma que possui o padrão 1-0 como antecedente e 2-0 como consequente, e outra onde 1-1 é o antecedente e 2-0 é o consequente. Estas informações podem ser confirmadas se olharmos na Figura 5.4, onde é possível ver que o padrão 2-0 é uma composição dos padrões 1-0 e 1-1.

```
2-0 1-0 1-1
```

Figura 5.5: Linha do arquivo de pais-e-filhos dos padrões

Somente estas regras de um passo são de interesse, pois como em um dos objetivos se pretende oferecer ao usuário um auxílio interativo na construção de seu modelo de processo, as regras não devem ser complexas. Regras de um passo são facilmente analisadas pelo usuário e este pode mais rapidamente encontrar aquela que casa com o seu desejo. Se as regras contivessem mais de um passo, um número muito maior de regras seria gerado para o usuário analisar, bem como este teria de analisá-las muito mais criteriosamente para verificar se contemplam todos os próximos passos que ele deseja inserir em seu modelo. Utilizando regras de um passo e uma modelagem

interativa este processo fica mais simples. Este fato representa um importante filtro para as regras geradas ao diminuir a quantidade destas que devem ser trabalhadas.

Na área de saídas (*Outputs*) da Figura 5.3, são listados os tamanhos de todos os padrões encontrados, o número de candidatos gerados e quantos destes candidatos realmente são padrões frequentes. Uma observação interessante e que chama a atenção é que o FSG foi capaz de encontrar um padrão frequente de tamanho 15, ou seja, pelo menos três processos possuem exatamente os mesmos 15 padrões de atividade conectados da mesma maneira. Utilizando-se o *TID-List File*, arquivo de saída que informa quais modelos contêm um determinado padrão, descobrimos que são exatamente 3 processos que contêm estes 15 padrões conectados da mesma maneira. Estes processos são: "Processo de Aprovação de Expiração de Textos", "Processo de Aprovação de Documentos" e "Processo de Aprovação de Expiração de Documentos". Basicamente, eles representam um processo onde inicialmente é buscado um responsável para realizar uma avaliação de um documento a qual pode ser igualmente submetida a um gestor para aprovação, mas também pode ser submetida a um novo autor. Além disto, esta avaliação pode ser prorrogada ou expirada. Esta associação característica entre os padrões de atividade pode também ser vista como um padrão, de granularidade maior, que representa parte de um processo de aprovação de algo. Futuramente, esta informação pode tornar-se um *template*, isto é, um molde, para a modelagem de processos de aprovação. Mais padrões minerados deste tipo serão vistos no Capítulo 6.

Por fim, o FSG exibe em sua saída apresentada na Figura 5.3 que gerou 10.163 subgrafos candidatos a padrões, sendo que 8.178 deles são padrões que atingem o suporte mínimo de três transações. É importante relembrar que esta geração de candidatos pelo FSG é otimizada e alternativas inválidas já são podadas em estágios iniciais tão cedo quanto possível. Durante a execução do algoritmo os candidatos a padrões de tamanho *k+1* são gerados através da união de dois padrões de tamanho *k* que possuem o mesmo subgrafo conexo de tamanho *k-1*, onde o tamanho de cada grafo é dado pela sua quantidade de arestas. Apesar de exibido na saída, o número de candidatos gerados não é de interesse para o presente estudo e presta-se mais a análises de desempenho do algoritmo as quais não são realizadas aqui.

Após esta execução, foi realizada uma nova solicitando-se ao algoritmo que fossem gerados apenas os **padrões frequentes máximos**, ou seja, aqueles padrões que não estão contidos em nenhum outro (KURAMOCHI, 2004). A Figura 5.6 apresenta o sumário dos padrões frequentes encontrados.

A geração dos padrões frequentes máximos reduz ainda mais o escopo de análise dos resultados como pode ser visto na quantidade total de padrões frequentes encontrada de 291. Convém observar que todos os padrões encontrados anteriormente também estão representados nesta execução, porém, diferentes padrões menores acabam sendo condensados em um único padrão maior. Todos os padrões pequenos encontrados na primeira execução que estão contidos em um padrão maior, são representados na saída desta execução apenas por este padrão maior. Uma visão clara deste fato está nos padrões de tamanho 10 a 15. Na primeira execução foram retornados 2227 padrões frequentes nesta faixa, enquanto que na segunda execução apenas um padrão foi retornado nesta faixa. Os 2227 padrões exibidos anteriormente não foram esquecidos na segunda execução, mas estão todos eles contidos no único padrão frequente de tamanho 15.

```
Size Frequent Patterns
1    10
2    55
3    128
4    59
5    13
6    8
7    5
8    7
9    5
10   0
11   0
12   0
13   0
14   0
15   1

Largest Frequent Pattern Size:               15
Total Number of Frequent Patterns Found:     291
```

Figura 5.6: Padrões frequentes máximos encontrados

Analisando-se o *TID-List File* verificou-se que os mesmos três modelos que contêm o padrão de tamanho 15 contêm todos os padrões de tamanho 10 a 15 retornados na execução anterior. Isto indica que apenas em três dos processos minerados os padrões de tamanho maior do que 10 são encontrados. Tal fato confere um suporte baixo para padrões deste tamanho, revelando que regras envolvendo estes padrões dificilmente serão as regras procuradas por um usuário no momento da modelagem. O próximo Capítulo realiza maiores análises sobre os dados minerados.

6 ANÁLISE DO CONHECIMENTO EXTRAÍDO

No Capítulo 5 algumas análises superficiais sobre os resultados da etapa de mineração de dados já foram realizadas. Neste Capítulo, diferentes análises serão executadas tendo por base os resultados desta etapa anteriormente executada.

Serão contempladas tanto análises manuais, quanto análises com o auxílio de *software* para automatizar parte do processo. Os padrões retornados na etapa de mineração de dados serão vasculhados e constatações interessantes serão apresentadas. Através das associações recorrentes serão buscados padrões de granularidade maior, ou seja, associações entre padrões de atividade que apareçam em um grande número de processos e que podem ter uma descrição em alto nível do que estão representando.

Para a realização de análises com o auxílio de *software* será desenvolvida uma aplicação capaz de simular a construção de modelos de processo com o auxílio de sugestões geradas a partir das associações recorrentes mineradas. O que se pretende fazer é tentar prever o futuro com base em informações extraídas a partir de casos anteriores. Este *software* contemplará um motor de inferência responsável por analisar um modelo de processo parcial e gerar sugestões de como este modelo deve evoluir, ou seja, quais padrões devem ser inseridos e onde.

Também serão realizadas análises possíveis através da modificação do *software* desenvolvido. Estas análises buscarão meios de melhorar os resultados obtidos de modo que a etapa de modelagem possa obter benefícios através da utilização de reuso de componentes.

Por fim, a ferramenta desenvolvida será utilizada para estudar como a segmentação de processos conforme determinada característica pode afetar os resultados da modelagem com a sugestão automatizada de padrões. O grupo de processos será dividido e cada grupo será submetido à mineração de seus padrões específicos. Em seguida, estes padrões específicos de cada grupo serão utilizados pela ferramenta para simular a construção de processos deste grupo e verificar se existe alguma vantagem nesta abordagem.

6.1 Análise de Co-ocorrências entre Padrões de Atividade

Através de uma analise manual das co-ocorrências encontradas pelo algoritmo de mineração de dados pretende-se encontrar padrões de granularidade maior do que os padrões base da pesquisa presentes em Thom (2009). O primeiro exemplo de padrões deste tipo foi descrito na Seção 5.2, onde se verificou a existência de uma associação

característica de tamanho 15 presente em três processos diferentes. Como analisado anteriormente, estes processos basicamente são referentes a processos de aprovação com a seleção de um avaliador e as possibilidades de se prorrogar o prazo para a aprovação ou cancelá-la quando o prazo se expirar. Além disso, o objeto a ser aprovado pode ser passado para avaliação em outro nível da hierarquia ou ser devolvido ao autor para modificações. Este padrão de granularidade maior pode ser descrito pela Figura 6.1 e pode ser entendido como um padrão que serve para representar uma aprovação dentro de uma organização hierárquica com um prazo especificado. Desta maneira, a associação recorrente entre os padrões de atividade conforme apresentada na Figura 6.1 pode ser entendida como um *template*, isto é, um esqueleto pré-definido, para a modelagem de processos que queiram descrever atividades de um processo deste tipo.

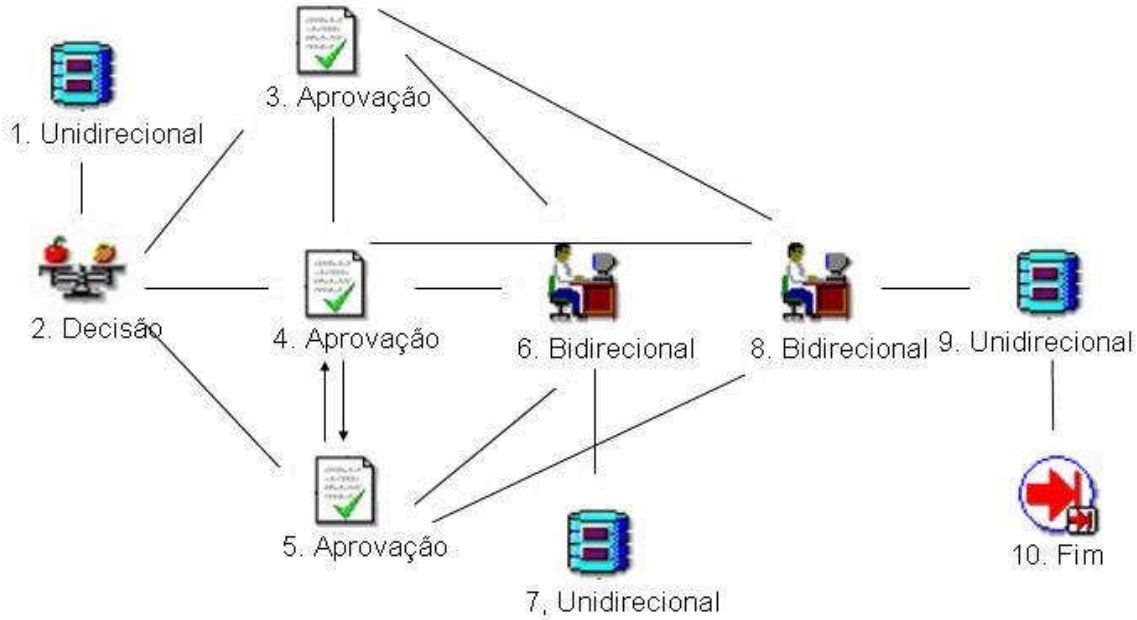

Figura 6.1: Padrão de aprovação em organização hierárquica com prazo

Os padrões de atividade identificados como 1 e 2 na Figura 6.1 podem ser vistos como atividades para se decidir quem aprovará o objeto alvo. Os padrões 3, 4 e 5 representam os diferentes níveis de aprovação e suas conexões demonstram como esta aprovação pode passar de um nível a outro. Os padrões 6 e 7 podem ser interpretados como uma saída da aprovação tal como um prazo expirado, por exemplo. Os padrões 8, 9 e 10 representam outra saída da aprovação, como uma aprovação bem sucedida, por exemplo, o que encerra o processo.

Para encontrar mais recorrências interessantes como esta, os padrões máximos foram analisados. Utilizando-se esta abordagem garante-se que os maiores padrões serão encontrados, já que todos os padrões menores que estão contidos em um mesmo padrão máximo são representados apenas por este na saída do algoritmo de mineração. Desta maneira espera-se encontrar quais são as maiores associações recorrentes entre os padrões de atividade, isto é, aquelas que contêm o maior número de padrões de atividade conectados uns com os outros. Esta abordagem reduz o espaço de análise dos resultados e torna esta tarefa mais fácil de ser realizada manualmente ao exibir uma menor quantidade de informações como saída. Além disso, analisando-se diretamente os padrões máximos tem-se a chance de encontrar uma associação recorrente que constitua

um processo completo, isto é, conexões entre os padrões de atividade que modelem um processo inteiro, evidenciando, então, um processo padrão ou, um *template*.

Verificando-se os padrões máximos com o maior suporte, uma descoberta interessante é que existem três deles que são encontrados em sete processos. A Figura 6.2 apresenta estas três co-ocorrências. O fato de elas serem encontradas como padrões nos processos minerados dá fortes evidências de que estas associações podem também vir a ser encontradas em novos processos e, portanto, podem representar padrões de granularidade maior.

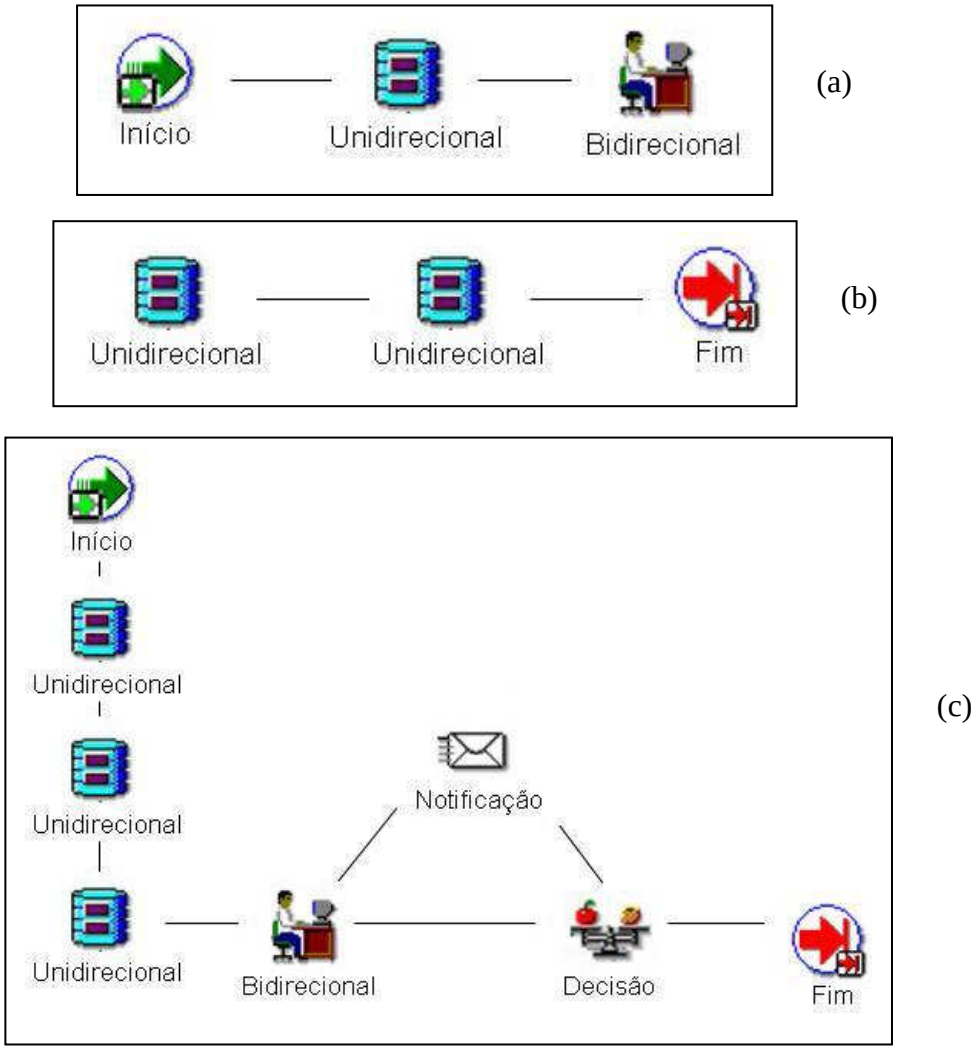

Figura 6.2: Padrões máximos encontradas em sete processos

A co-ocorrência da Figura 6.2 (a) representa um modelo com o seu nó de início, conectado com um padrão unidirecional que está conectado com um padrão bi-direcional. Ela é encontrada nos seguintes processos: “Aprovação de cancelamento de documentos”, “Processo de alteração de layout”, “Feedback de colaboradores: Processo principal”, “Feedback de colaboradores: Realizar primeira reunião”, “Concessão de acesso a transação adicional”, “Declaração de ciência de normas” e “Devolução do adiantamento ao financeiro”. Esta co-ocorrência encontrada nos processos minerados indica que, em uma tentativa de se predizer como um novo processo sendo modelado será iniciado, a maior probabilidade de acerto é sugerir que será conforme esta co-

ocorrência. Esta pode então ser vista como um padrão para início de processo, indicando que ao ser iniciado, uma atividade unidirecional é executada e, logo em seguida, uma atividade bi-direcional. A atividade unidirecional pode ser entendida como um disparo de uma atividade não relacionada com a próxima de modo que, quando o processo ficar bloqueado na espera da resposta da atividade bidirecional, algo esteja sendo processado, ou executado, isto é, a atividade unidirecional.

A Figura 6.2 (b) representa uma co-ocorrência onde um final normal de processo é precedido por dois padrões unidirecionais encadeados. Ela é encontrada nos seguintes processos: “Processo Principal: Aprovação de eventos”, “Sub-Processo Publicação e Distribuição de Documento”, “Sub-Processo Busca de Dados Necessários”, “Criação Revisão Documentos”, “Aprovação do caixa da sede”, “Solicitação de afastamento” e “Registro de ocorrência”. Esta recorrência pode ser interpretada como um padrão de fim de processo e pode ser utilizada por um motor de inferência para sugerir a um usuário que está modelando seu processo de negócio que um nó de fim normal seja inserido no modelo quando dois padrões unidirecionais estão em sequência, por exemplo. Este encadeamento que precede o fim pode ser entendido como atividades de conclusão da execução do processo, etapas para a finalização deste, ou arremates necessários para o fim normal do processo.

A recorrência da Figura 6.2 (c) é encontrada nos seguintes processos: “Sub-Processo Envio e Recebimento de Eventos de Busca de Metadados”, “Sub-Processo Envio e Recebimento de Eventos de Cancelamento de Publicação”, “Sub-Processo Envio e Recebimento de Eventos de Inclusão de Autor”, “Sub-Processo Envio e Recebimento de Eventos de Prorrogação de Descarte”, “Sub-Processo Envio e Recebimento de Eventos de Revogamento de Documento”, “Sub-Processo Envio e Recebimento de Eventos de Expiração de Documento” e “Sub-Processo Envio e Recebimento de Eventos de Prorrogação de Vigência”. Esta recorrência é a mais interessante, pois representa um modelo completo e todas suas ocorrências foram encontradas em modelos de uma companhia de gerenciamento de conteúdo. Como se pode ver pelos próprios títulos dos processos, todos eles, sem exceção, representam sub-processos de envio e recebimento de algo. Esta análise permite a identificação de um padrão para processos de envio e recebimento. Desta maneira, caso seja necessário a modelagem de um novo processo deste tipo, pode-se já partir desta base. Além disto, caso existam outros processos de envio e recebimento, estes podem ser analisados para verificar se estão corretos, ou se deveriam ser modelados conforme estes sete encontrados, já que esta informação descoberta se caracteriza uma recorrência.

Estas observações sobre os padrões máximos são bastante interessantes e motivadoras. Em vista disto, os padrões com o maior suporte também foram analisados. Neste caso, todos os padrões foram minerados, e não somente os padrões máximos. Alguns dos mais destacados estão representados na Figura 6.3.

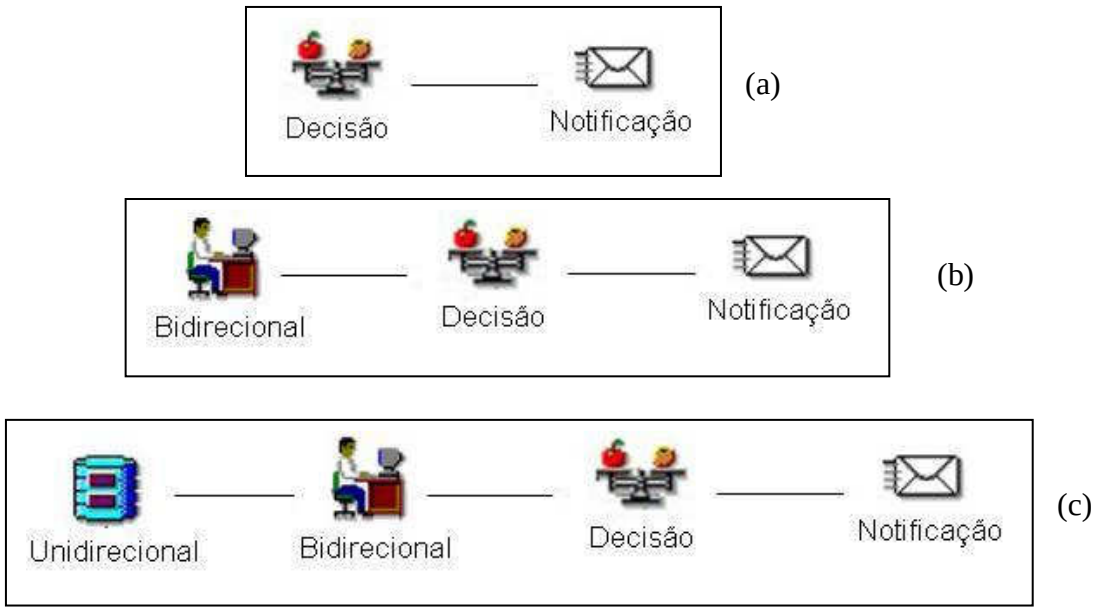

Figura 6.3: Padrões com os maiores suportes

A Figura 6.3 (a) apresenta o padrão com o maior suporte encontrado. Como era de se esperar, ele possui tamanho um, já que quanto menor o padrão, mais facilmente ele pode ser encontrado em diferentes processos. Esta recorrência foi encontrada em mais de 40% dos processos minerados. Uma interpretação que pode ser dada a este padrão é de que após uma decisão, há uma notificação, muito provavelmente da própria decisão tomada às partes interessadas; ou, interpretando-se pelo outro sentido da aresta, que normalmente antes de uma decisão existe uma notificação.

A Figura 6.3 (b) representa o padrão de tamanho dois com o maior suporte encontrado. Ele pode ser visto em 20% dos processos minerados. Uma observação detalhada indica que este padrão é uma evolução do padrão apresentado na Figura 6.3 (a). Ele possui os mesmos padrões de decisão e notificação, acrescentando um padrão bidirecional conectado ao padrão de decisão. Uma interpretação possível para este padrão é de que antes de uma atividade de decisão, normalmente é necessário realizar alguma atividade prévia, a qual deve ser concluída antes que a decisão seja tomada. Da mesma maneira que na Figura 6.3 (a), após a decisão ser tomada, acontece uma notificação. Lembrando que esta é apenas uma interpretação possível para a associação recorrente minerada, visto que diferentes interpretações podem ser dadas se as arestas forem consideradas em diferentes sentidos. O fato de se utilizar padrões não dirigidos permite esta flexibilidade, deixando que o usuário defina a ordem das arestas de modo a respeitar a semântica desejada para o seu processo sendo modelado.

Na Figura 6.3 (c) tem-se novamente uma evolução do padrão apresentado no item anterior da Figura 6.3. Ele aparece em 12,8% dos processos minerados e acrescenta um padrão unidirecional na associação. Este padrão unidirecional pode ser visto como uma atividade que deve ser disparada antes da atividade bidirecional que leva à tomada de decisão se considerar-se a interpretação já realizada sobre o item (b).

Todas estas associações recorrentes podem ser vistas como padrões de maior granularidade, pois são compostas por padrões de atividade e possuem uma semântica maior, aparecendo em uma grande quantidade de processos. Entretanto, neste ponto, nota-se a dificuldade em analisar manualmente todos os padrões minerados, já que são

mais de 8000. Em virtude disto, a Seção 6.2 introduz a ferramenta que foi desenvolvida para realizar algumas análises de forma automatizada e retornar informações interessantes que permitam que se faça uma análise do conhecimento descoberto.

6.2 Uma Ferramenta para a Mineração de Modelos de Processos

Para tornar a etapa de avaliação dos padrões minerados mais fácil de ser realizada foi desenvolvido um programa capaz de automatizar esta fase. A ideia é que, a partir de um conjunto de padrões minerados, a ferramenta seja capaz de automatizar algumas análises e retornar resultados consolidados que podem dar indicações da utilidade e qualidade do conhecimento descoberto. Além de automatizar a etapa de verificação dos resultados, a ferramenta foi desenvolvida de maneira a auxiliar em parte da etapa de mineração de dados.

Devido à característica do algoritmo de mineração de grafos utilizado, as ocorrências dos padrões frequentes dentro dos processos são contadas apenas uma vez, independentemente se estas aparecem uma ou dez vezes dentro de um mesmo modelo de processo. Por exemplo, no processo apresentado na Figura 6.4 têm-se duas ocasiões em que um padrão de aprovação segue um padrão de decisão. O FSG conta estas duas ocorrências dentro do mesmo processo apenas uma vez. No entanto, é interessante que todas as ocorrências sejam levadas em consideração para que realmente se tenha uma medida da recorrência das associações entre os padrões de atividades dentro dos processos e não somente uma contagem por processo. Por este motivo, a ferramenta desenvolvida realiza um pós-processamento após a execução do FSG para obter estes números. Ainda dentro da etapa de mineração de dados, tendo o FSG retornado os padrões frequentes, a ferramenta é responsável por atualizar a frequência destes. Ou seja, para cada padrão frequente retornado pelo FSG, os processos que contêm este padrão frequente são automaticamente analisados e é verificado quantas vezes o padrão frequente encontrado está presente em cada modelo. De posse destes resultados, a frequência de cada padrão frequente é atualizada para representar a contagem real de ocorrências e não por processo.

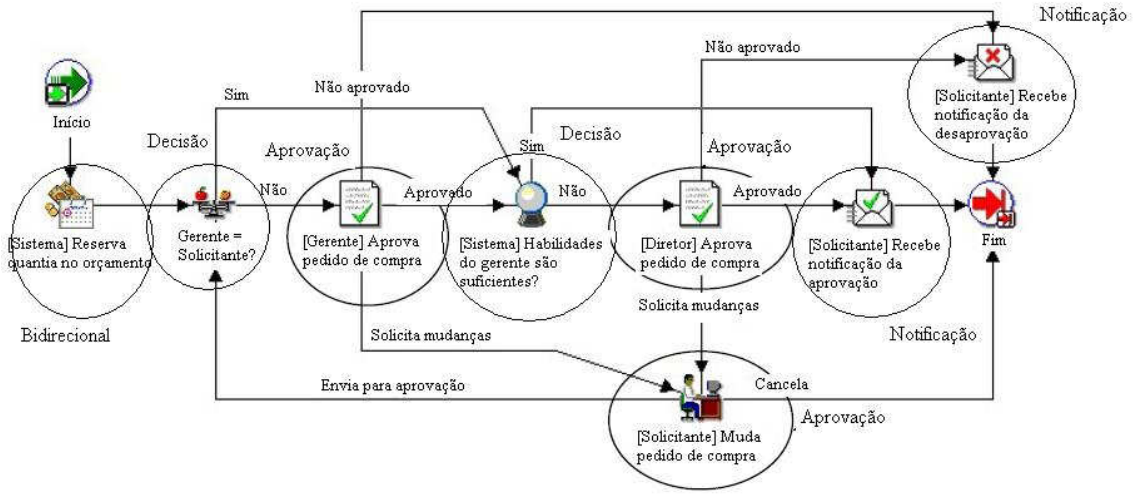

Figura 6.4: Processo de pedido de compra

Além do pós-processamento da etapa da mineração de dados a ferramenta desenvolvida contempla um motor de inferência. A partir dos padrões frequentes encontrados na etapa de mineração de dados, dado um modelo de processo parcialmente

desenhado, este motor é capaz de sugerir uma lista de quais são as estruturas mais prováveis que devem ser inseridas no modelo parcial para a continuidade de sua modelagem. Em outras palavras, o motor de inferência baseia-se nos padrões recorrentes encontrados para tentar predizer como o modelo em questão sendo desenhado irá evoluir.

Paralelamente ao motor de inferência foi desenvolvido um módulo capaz de integrar-se a ele e simular a construção de uma série de modelos de processos passados como entrada. Esta simulação de construção dos modelos é realizada como se um usuário estivesse desenhando o modelo de processo com o auxílio das sugestões oferecidas pelo motor de inferência. Este módulo de testes é o responsável por realizar as computações necessárias para imprimir as informações que serão verificadas para avaliar a qualidade dos padrões minerados.

A Figura 6.5 ilustra a ferramenta desenvolvida e as interações dos seus principais componentes em alto nível. Pode-se ver que ela recebe um conjunto de modelos de processos como entrada e então realiza uma mineração de dados. Após, obtém-se um conjunto de co-ocorrências dos padrões de atividade considerando-se a sua contagem de ocorrências geral. Estes padrões frequentes, juntamente com um modelo parcialmente desenhado são a entrada para o motor de inferência. A execução deste motor de inferência retorna uma lista de recomendações contendo os prováveis padrões de atividade a serem inseridos no modelo e onde estes devem ser inseridos para continuar a modelagem do processo de negócio parcial em questão. O módulo de testes realiza a chamada deste motor de inferência e é capaz de simular um usuário modelando um processo de negócio passado como entrada. Com isto, ele é capaz de avaliar automaticamente como as sugestões da lista gerada pelo motor de inferência poderiam ser utilizadas. A saída do módulo de testes é uma série de contagens realizadas durante este processo de simulação para que os padrões minerados então sejam avaliados e seja verificado se pode haver algum ganho através da sua utilização.

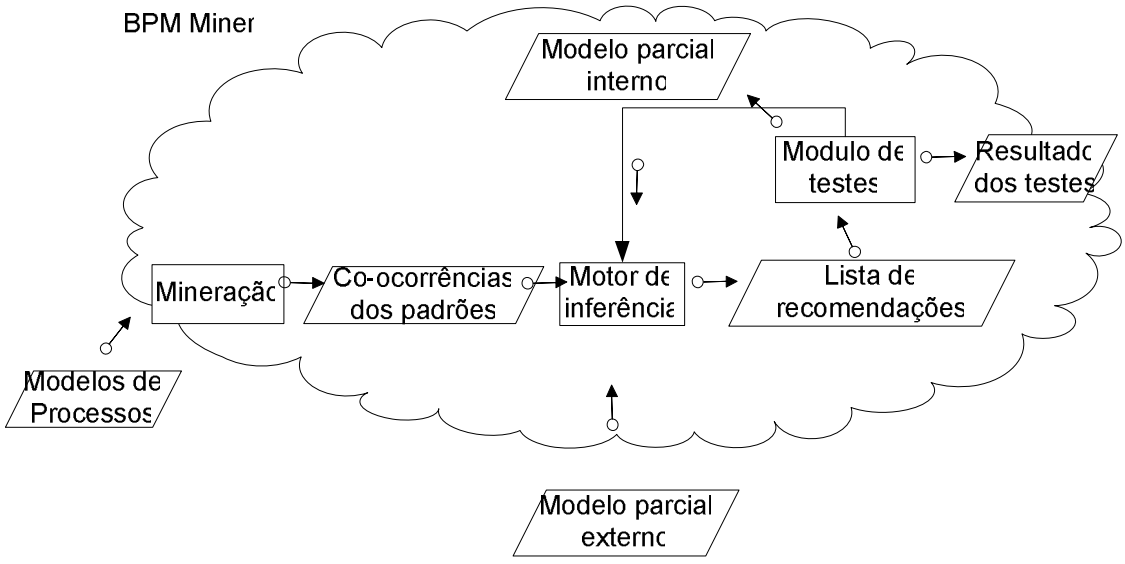

Figura 6.5: BPM Miner

A ferramenta recebeu o nome de BPM Miner e foi desenvolvida utilizando-se uma estrutura orientada a objetos onde diversas classes, cada uma com suas responsabilidades específicas, foram implementadas. Para isto, a linguagem de programação Java foi utilizada no ambiente de desenvolvimento Eclipse. A Figura 6.6

ilustra as principais classes da ferramenta elaborada através de um diagrama de sequência.

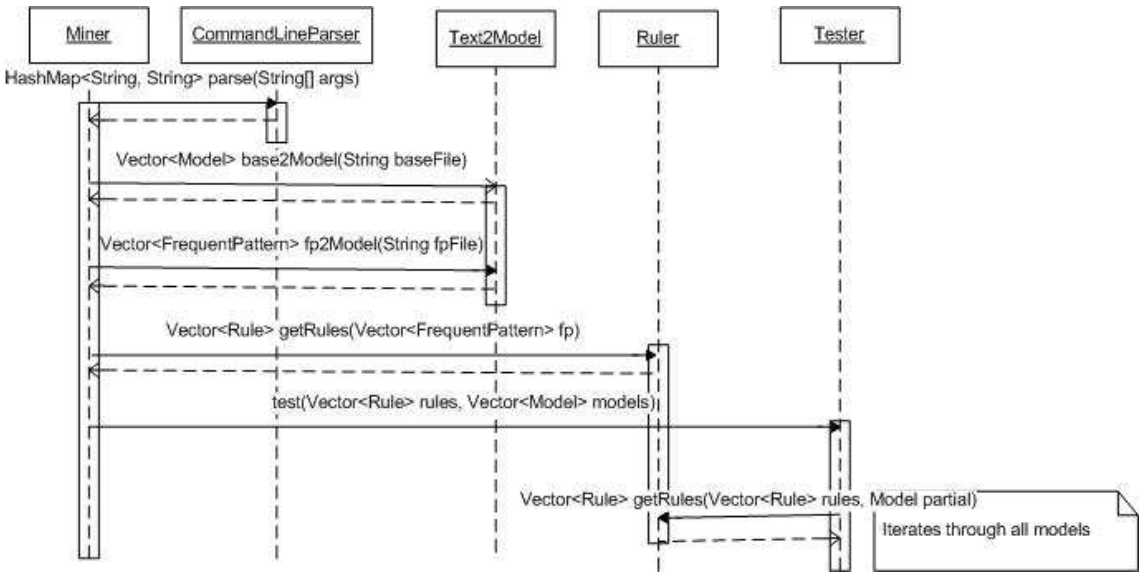

Figura 6.6: Principais classes do BPM Miner

A classe Miner é a principal classe do sistema. Ela é responsável por rodar o módulo de testes sobre uma dada base de padrões frequentes passando um conjunto de modelos de processos para serem testados. Para isto, ela deve receber como parâmetros um arquivo que contém os padrões frequentes, um arquivo que contém os relacionamentos de pai-filho destes padrões e o arquivo que contém os modelos de processos cuja construção será simulada com o auxílio do módulo de sugestões automáticas. O resultado da simulação da construção dos modelos de entrada é impresso na saída padrão e pode facilmente ser redirecionado para algum arquivo caso o usuário deseje.

Recebendo seus parâmetros, a classe Miner então constrói os grafos dos processos que serão simulados realizando um *parsing* sobre o arquivo de entrada. Os outros dois arquivos são utilizados para construir o conjunto dos padrões frequentes, juntamente com as suas frequências, e uma lista com os identificadores dos padrões que são diretamente seus filhos.

dada a regra de associação $X \Rightarrow Y$

X implica Y

define-se

$$suporte = \frac{\text{Número de registros com } X \text{ e } Y}{\text{Número total de registros}}$$

$$confiança = \frac{\text{Número de registros com } X \text{ e } Y}{\text{Número de registros com } X}$$

Figura 6.7: Fórmulas para cálculo de suporte e confiança

Tendo lido os modelos de processos e os padrões frequentes de entrada, a classe Miner então invoca a classe Ruler. Esta classe possui um método responsável por computar todas as regras de associação com seus respectivos valores de suporte e

confiança. Os valores de suporte e confiança das regras podem facilmente ser obtidos a partir dos valores de suporte de seu antecedente e consequente como ilustram as fórmulas da Figura 6.7.

Filtros genéricos na geração das regras podem ser aplicados no método responsável por gerar todas as regras na classe Ruler. Por este motivo, neste passo são geradas somente as regras de um passo, ou seja, onde o antecedente e o consequente diferem por apenas uma aresta e, possivelmente, um nodo como será visto a seguir. Este tipo de filtro é genérico, pois não está considerando nenhum modelo parcialmente desenvolvido para gerar as regras de associação, mas sim, considera um aspecto global ao qual todas as regras geradas devem obedecer.

Uma regra com um suporte S% indica que em S% dos modelos de processos minerados, os padrões frequentes X e Y foram encontrados. Como X está contido em Y, já que este é o padrão X acrescido de uma aresta e, possivelmente um nodo, o número de processos onde X e Y são encontrados é o mesmo que o número de modelos onde Y foi encontrado. Uma regra com uma confiança C% indica que em C% dos modelos de processos minerados, sempre que o subgrafo frequente X foi encontrado, o subgrafo Y também estava presente. Novamente, como Y é o padrão X acrescido de uma aresta e, possivelmente um nodo, pode-se dizer que em C% dos casos que encontramos X, existe a adição de uma estrutura que o transforma em Y e daí a informação necessária para fazer a predição.

Na geração das regras de associação, a lista de pais-filhos dos padrões frequentes é de fundamental importância ao indicar prontamente todos os filhos imediatos de um padrão. Desta maneira, não há o gasto computacional de se verificar se um determinado padrão frequente está contido em um outro para então gerar uma regra de associação que evolui do primeiro para o segundo. Basta que sejam geradas regras que evoluem dos padrões filhos para os padrões pais e assim todas as regras de um passo estarão sendo geradas.

Após a geração inicial das regras, o módulo de simulação é invocado. Este módulo recebe como parâmetro uma lista de regras de associação e os modelos que deverão ter a sua construção simulada com base nestas regras. Ele itera sobre todos os modelos que recebe como entrada e simula a construção de cada um deles, aresta por aresta. Para dar início à simulação é construído o modelo inicial, o qual contém apenas a primeira aresta do modelo conforme informada no arquivo de entrada. Este é um passo que seria dado pelo usuário sem o auxílio do módulo de sugestões automáticas, pois, para gerar sugestões, o módulo baseia-se em um modelo parcialmente construído. Sem qualquer parte do processo já desenvolvida, é impossível realizar um casamento com alguma regra gerada e, por este motivo, a inserção da primeira aresta no modelo é realizada sem a geração de sugestões.

Tendo o modelo parcial inicial contendo apenas uma aresta, ele é submetido a um método da classe Ruler que recebe um modelo parcialmente construído e um conjunto de regras e retorna uma lista ordenada de regras que são aplicáveis para dar continuidade ao desenho deste. Para identificar se uma regra é aplicável a um modelo, é necessário verificar se o antecedente da regra está contido nele. Desta forma, todas as regras retornadas estarão indicando uma possível evolução para o modelo parcialmente desenvolvido. Esta evolução considera parte do modelo e insere uma nova aresta neste, possivelmente com a inserção de um novo nodo também caso seja necessário. É utilizada esta abordagem de crescimento por aresta, pois nos casos em que o usuário

desenharia um ciclo em seu modelo, nenhum nodo seria inserido; a aresta do ciclo seria apenas criada entre os nodos já existentes. Caso uma nova aresta a ser inserida aponte para um nodo ainda inexistente, este também é automaticamente inserido ao modelo, dando a sua evolução.

A ordenação das regras na lista de sugestões é dada pelo método de ordenação implementado na classe Rule, a ser exibida no diagrama da Figura 6.12. Este método ordena as regras com base em dois critérios: o primeiro é com base no tamanho da regra; o segundo com base na confiança. Desta maneira, a lista de sugestões apresentará as regras em ordem decrescente de tamanho e, para regras do mesmo tamanho, em ordem decrescente de confiança. Assim, são oferecidas primeiramente as regras que casam com a maior parte do processo parcialmente desenhado. O objetivo desta abordagem é utilizar a maior quantidade de informação possível para tentar prever a evolução do desenho de um processo de negócio. O próximo passo após as regras aplicáveis terem sido selecionadas para a lista de sugestões é procurar nesta lista aquela regra que apresenta a evolução do modelo que seria escolhida pelo usuário conforme dado no arquivo de entrada, ou seja, a regra que possui como antecedente o modelo apenas com a sua primeira aresta, e. no consequente, o modelo com a primeira e segunda arestas especificadas. Assim, conforme definido no arquivo de entrada, a ferramenta simula um usuário desenhando os modelos de processos.

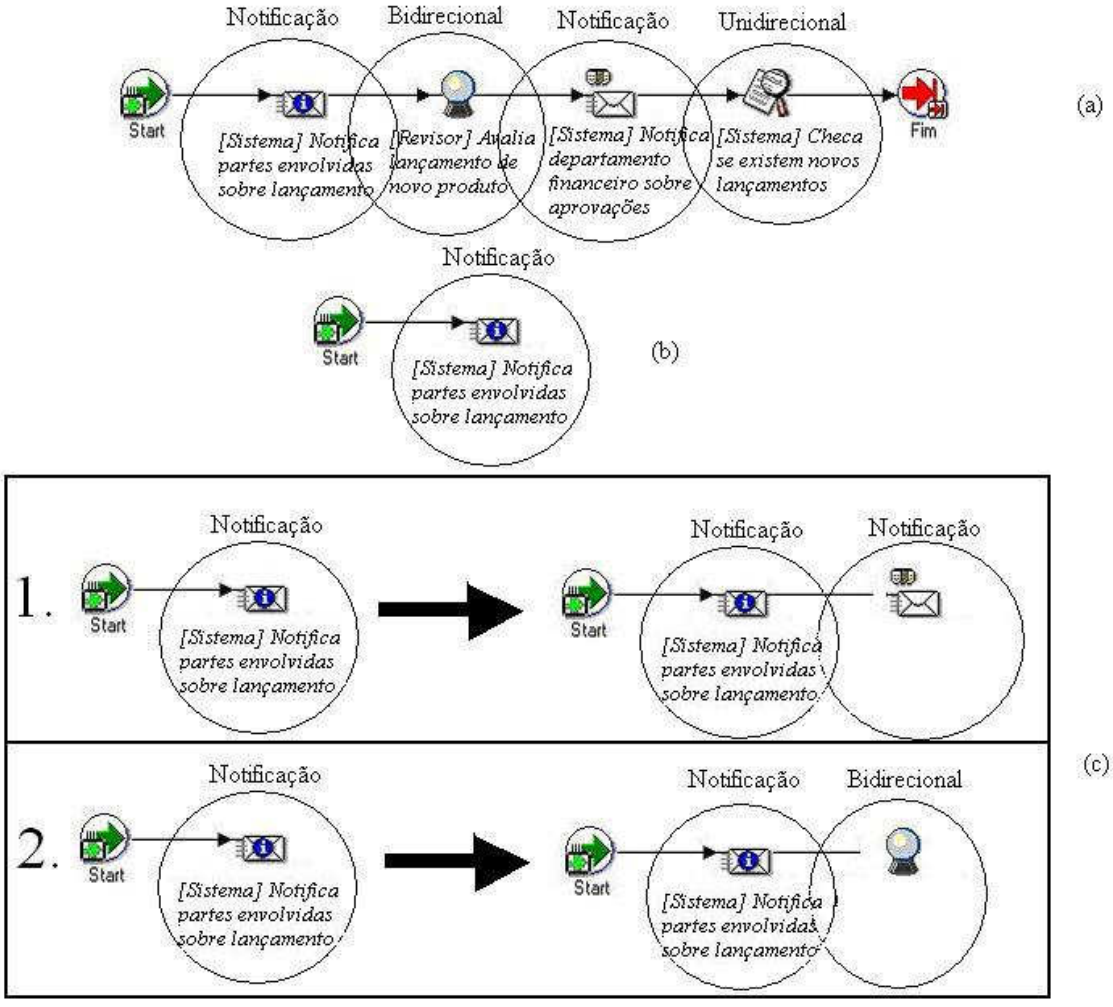

Figura 6.8: Exemplo de evolução da modelagem

A Figura 6.8 apresenta um exemplo do funcionamento do módulo de testes. No item (a) está apresentado o modelo completo que está tendo a sua construção simulada. O item (b) apresenta o modelo parcial desenhado até um determinado momento da simulação. O item (c) apresenta possíveis regras que podem ser geradas pelo motor de inferência ao receber o modelo do item (b) como entrada e um certo conjunto de padrões freqüentes. Neste caso, o módulo de testes escolheria como regra para continuar o desenho do modelo parcial a número 2, já que ela representa a conexão de um padrão bidirecional ao padrão de notificação já existente, o que está de acordo com o modelo apresentado no item (a) da figura.

Se a evolução desejada para o modelo é encontrada na lista de sugestões este acerto é armazenado. Senão, um erro é contado. Após isto, o modelo parcial é ampliado para conter a segunda aresta do arquivo de entrada e o mesmo processo é repetido novamente até que a inserção de todas as arestas tenha sido simulada.

Uma observação que deve ser feita sobre esta simulação é o fato de que em um caso de uso real da lista de sugestões não se pode prever o comportamento do usuário, isto é, as arestas não são inseridas em um modelo sempre na mesma ordem. Principalmente com o auxílio da lista de sugestões. A Figura 6.9 ilustra um exemplo onde é possível que em determinados momentos um usuário opte por primeiro modelar o ramo de cima do modelo, enquanto que em outros o primeiro ramo a ser simulado seja o de baixo. Este não determinismo não é simulado nos testes, sendo considerado que a escolha do usuário deve ser especificada conforme a ordem das arestas no arquivo de entrada. Uma alternativa para simular opções diferentes do usuário pode ser feita com diferentes ordens das arestas no arquivo de entrada.

Figura 6.9: Exemplo de modelo de processo com dois ramos

Um grafo que modela o ramo *não* primeiro seria definido conforme Figura 6.10 (a), enquanto que um grafo que modelo o ramo *sim* primeiro seria definido conforme Figura 6.10 (b). Note a inversão no ordenamento dos nodos que representam os padrões de notificação e bidirecional.

T # Modelo Exemplo
V 0 S
V 1 D
V 2 N
V 3 B
U 0 1 N
U 1 2 N
U 1 3 N
(a)

T # Modelo Exemplo
V 0 S
V 1 D
V 2 B
V 3 N
U 0 1 N
U 1 2 N
U 1 3 N
(b)

Figura 6.10: Diferentes ordens do mesmo modelo

Para a execução da simulação de construção dos modelos uma heurística ainda é necessária. Esta se faz exigida, pois existem casos em que é não é possível determinar de forma automática se um certo consequente de uma regra já está presente ou não no modelo. Isto é importante, porque se o consequente já existir no modelo parcialmente desenvolvido regras incorretas podem acusar um *matching*. A Figura 6.11 ilustra um exemplo de um *match* que seria incorreto para um certo modelo.

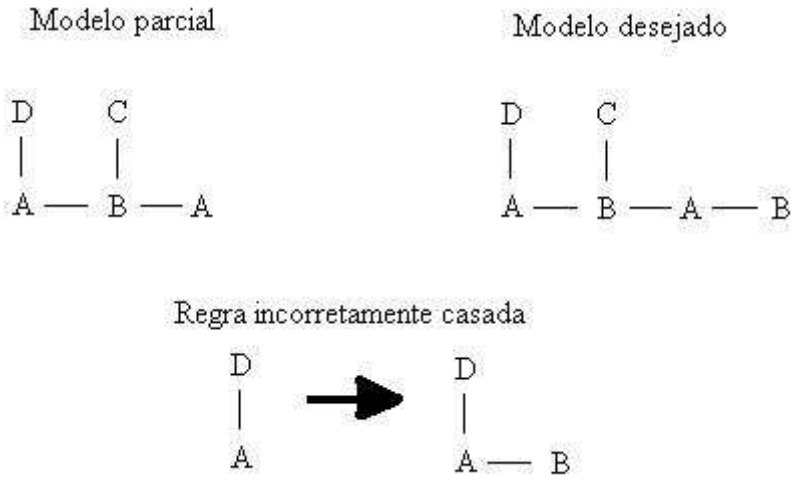

Figura 6.11: *Matching* incorreto de uma regra

Conforme a Figura 6.11, a regra correta deveria indicar a inserção de um nodo B conectado ao nodo A. Contudo, uma regra pode ser incorretamente casada. Se for verificado apenas se o antecedente da regra está contido no modelo parcial, e se o consequente da mesma está contido no modelo desejado, resultados errados podem surgir. Para evitar este tipo de erro a heurística adotada verifica se o consequente das regras a serem testadas já está contido no modelo antes de a aresta desejada ser inserida. Casos como este acusariam um *matching* incorreto na fase de verificação e devem ser eliminados. A alternativa adotada para eliminá-los consiste em, no momento de verificação do casamento das regras, remover temporariamente do modelo parcialmente desenvolvido a penúltima aresta inserida e o restante do modelo que ficar desconexo

através da remoção desta aresta caso algum consequente de alguma regra já esteja presente no modelo parcial. Este procedimento é realizado para a penúltima aresta e todas as arestas inseridas antes dela até que nenhum consequente das regras a serem testadas esteja presente no modelo parcialmente desenvolvido antes da inserção da aresta desejada. Esta heurística limita o tamanho dos modelos que devem ser analisados e restringe o espaço de busca das regras tornando ainda a execução do algoritmo mais rápida. Para eliminar esta heurística seria necessário a utilização de algoritmos de *matching* de nodos de grafos. Estes algoritmos são bastante complexos e semelhantes a algoritmos de determinação de isomorfismo de grafos, o que se tem acreditado estar na classe dos algoritmos NP-completos, conforme a Sub-Seção 4.3.1.

Na medida em que a simulação avança, a classe de testes armazena alguns números e realiza alguns cálculos e totalizações para exibi-los ao final do processamento. Como resultado, é gerado um relatório o qual é impresso na saída padrão. Este relatório contém diversas informações úteis para análise, tais como:

- número total de regras geradas, retornadas pela classe Ruler;
- número de vezes em que ocorreu a geração da lista de sugestões e escolha de uma regra a ser aplicada simulando o comportamento de um usuário, ou seja, o número de testes executados na simulação para verificar a qualidade da lista de sugestões;
- média e desvio padrão do tamanho das listas de sugestões geradas nos passos simulados,
- quantidade e porcentagem de regras utilizadas na modelagem conforme a sua posição na lista de sugestões, ou seja, o número de regras da posição *n* da lista que um usuário teria escolhido para continuar a modelagem de seu processo como desejado e o número de vezes em que a regra necessária não estava presente na lista e;
- a quantidade e porcentagem de regras utilizadas na modelagem conforme o seu tamanho, isto é, quantas regras que possuem um antecedente com *n* arestas teriam sido escolhidas por um usuário para continuar a modelagem de seu processo conforme desejado.

A Figura 6.12 ilustra um diagrama dos principais objetos e relacionamentos que são utilizados pelo BPM Miner. Cada objeto também contém as suas principais propriedades e métodos que foram implementados. Foi utilizada para auxílio na representação e implementação dos grafos a biblioteca jGraph (JGRAPH, 2001).

Conforme pode ser visto na Figura 6.12, a classe de nodos foi definida de modo a implementar a interface Comparator do Java para permitir a utilização de funções de ordenação nativas do Java em determinados métodos onde necessário. Dois nodos são comparados pelo seu tipo, ou seja, a função de comparação induz uma ordenação alfabética dos padrões. Possuir algum critério de ordenação é importante para otimizar a verificação se um padrão frequente está contido em um modelo, seguindo a mesma ideia utilizada pelo FSG e apresentada na Sub-Seção 4.3.1.

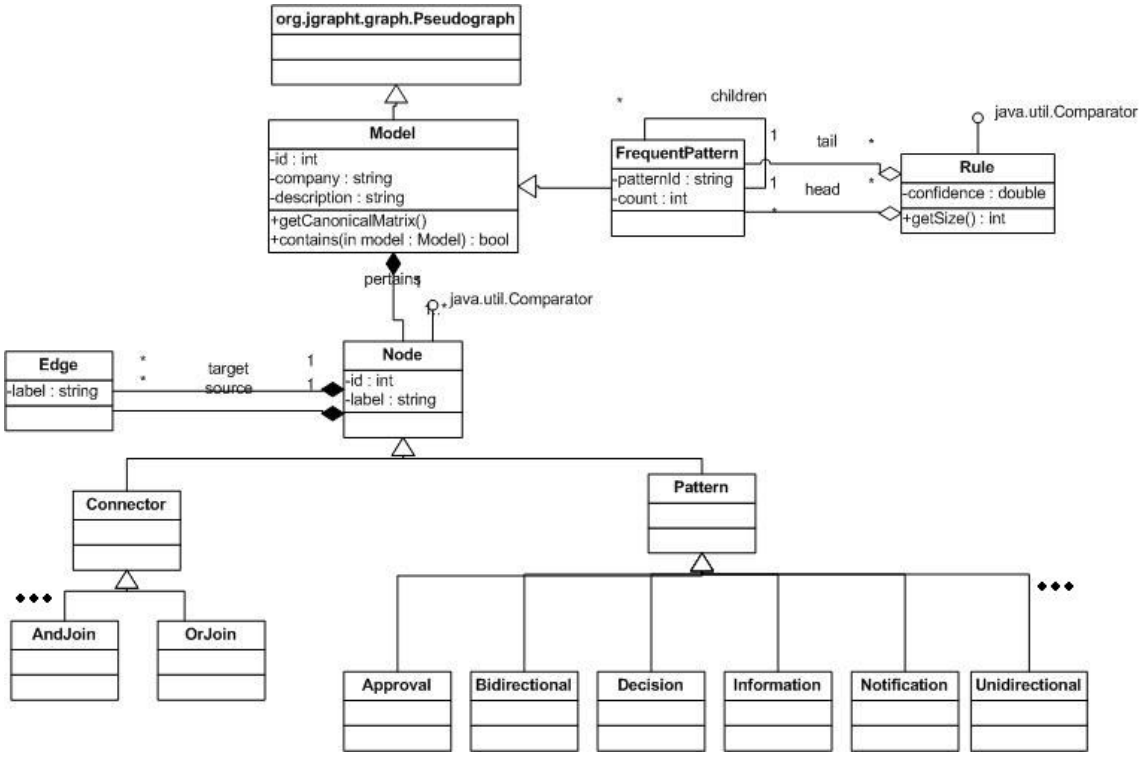

Figura 6.12: Objetos do BPM Miner

A classe que representa um modelo possui alguns métodos que são de extrema importância para o desenvolvimento dos módulos subsequentes. Dentre estes métodos importantes pode-se citar aquele que retorna a matriz canônica do grafo e aquele que verifica se um dado grafo está contido no modelo em questão. O cálculo da matriz canônica é realizado com a ajuda da técnica de invariantes de vértices, a qual também é utilizada no algoritmo FSG.

Foi visto que a técnica de invariantes de vértices consiste em encontrar características dos vértices de um grafo que permitam separá-los em partições ordenadas. Desta maneira, a matriz canônica deve obedecer a determinadas propriedades, o que agiliza o seu cálculo. O invariante de vértice utilizado nesta implementação é o tipo de padrão que o vértice representa. Esta propriedade é facilmente definida para qualquer vértice de um modelo e possui um ordenamento natural que é baseado na ordem alfabética dos nomes dos padrões. Assim, vértices que representam um mesmo padrão são agrupados na mesma partição.

O próximo passo para encontrar a matriz canônica é encontrar uma ordenação característica para os nodos pertencentes a uma mesma partição. A ordenação característica que é buscada neste caso é aquela que proporcione o maior código para aquela partição. O código de uma partição é obtido a partir da matriz de adjacência simplesmente concatenando-se as suas linhas. Como esta busca é feita exaustivamente por todas as possibilidades já que não existe fórmula para encontrá-la, é muito importante a utilização dos invariantes de vértices que reduzem bastante o escopo de busca. Uma vez encontrada a partição com o maior código, parte-se para a investigação da próxima até que não haja mais partições a serem verificadas. Ao final, as ordenações que possuem os maiores códigos em cada partição são retornadas para compor a matriz canônica. Este método garante que grafos iguais representados em diferentes ordens terão a mesma matriz canônica gerada e poderão ser identificados como idênticos. A

utilização apenas do padrão que um nodo representa como invariante mostrou-se suficiente para atingir bons tempos de execução. Entretanto, caso fosse necessário melhorar ainda mais este ponto poderiam ser utilizados outros invariantes tais como o grau de entrada e saída dos vértices, por exemplo.

Outro método importante existente na classe que representa um modelo é aquele que verifica se um modelo está contido em outro. Neste método a utilização da matriz canônica é fundamental para realizar a verificação necessária. O primeiro passo consiste em confirmar através do tamanho dos modelos que, de fato, é possível que um esteja contido no outro. Isto é, o modelo que deve estar contido não pode ser maior do que aquele no qual se verificará se ele realmente está contido. Em seguida deve-se verificar em uma lista *hash* se um dado modelo já foi testado e verificado que está contido no modelo em questão. Isto elimina o dispêndio de processamento para casos já analisados. Implementar uma lista *hash* para a verificação se um modelo não está contido na estrutura analisada não se fez útil, pois os modelos evoluem através da adição de novas arestas e novos nodos a cada iteração. Desta maneira, um modelo que não estava contido anteriormente pode passar a estar contido no modelo em passos seguintes. Ao utilizar a otimização para a verificação dos modelos que estão contidos em um grafo deve-se ter o cuidado de limpar a lista *hash* caso alguma aresta ou nodo seja removido do modelo, pois algumas estruturas podem passar a não estarem mais contidas no objeto analisado.

Passado este passo é realizada uma verificação se os modelos sendo testados são iguais. Esta verificação é mais rápida e, portanto, é realizada o quanto antes para evitar cálculos mais complexos.

O último passo consiste em testar todas as possibilidades restantes até que se confirme que um modelo está contido no outro, ou que todas as alternativas tenham sido esgotadas. Nesta etapa são construídos todos os subgrafos possíveis do modelo em questão que possuem nodos representando os mesmos padrões do modelo o qual se quer saber se está contido ou não. Desta maneira, cada subgrafo deve ser verificado se é igual ao modelo que deve estar contido.

Como visto na Figura 6.12 a classe que implementa um padrão frequente estende a classe que implementa um modelo, pois um padrão frequente é um modelo que possui a ele associado um certo número de ocorrências e uma lista de padrões frequentes filhos.

Já a modelagem da classe de regras a define de modo a implementar a interface Comparator do Java. Esta interface contempla um método para realizar a comparação de duas regras. Este método é implementado na classe Rule para ordenar regras conforme critérios ali estabelecidos. Desta maneira, a alteração da ordem como os elementos das listas de sugestões são apresentados ao usuário pode facilmente ser realizada caso necessário para diferentes estudos.

Ao longo da implementação do BPM Miner a tecnologia de controle de versões foi utilizada a fim de garantir um registro de todos os passos de evolução da aplicação, bem como permitir o gerenciamento das mudanças efetuadas e facilitar o acesso ao código fonte desenvolvido.

6.3 Aplicando a Ferramenta na Análise de Co-ocorrências

Após a implementação da ferramenta descrita na Seção 6.2, a mesma foi utilizada na análise das co-ocorrências encontradas nos processos minerados. Para iniciar esta

avaliação, os processos sofreram um particionamento estratificado aleatório. Isto quer dizer que os processos de cada organização foram randomicamente divididos em dois conjuntos: um conjunto de treinamento e outro de validação. O primeiro, corresponde a dois terços dos processos e é o conjunto onde os padrões frequentes são minerados. O segundo conjunto de processos, que corresponde ao terço restante, é onde os padrões minerados serão avaliados, isto é, o conjunto onde será verificado o quão preditivo o conhecimento minerado é capaz de ser. Realizando a avaliação desta maneira é possível garantir que o conhecimento minerado será verificado sobre modelos de processos "nunca vistos", o que elimina possíveis coincidências caso o conhecimento minerado fosse verificado sobre os próprios processos onde ele foi encontrado. A Figura 6.13 apresenta um resumo dos resultados retornados através da utilização da ferramenta desenvolvida com um conjunto de treinamento para minerar os processos e outro para validar o conhecimento extraído.

Pode-se observar que foram geradas 48850 regras a partir dos padrões frequentes encontrados. Na simulação da construção dos modelos do conjunto de validação com o auxílio dos padrões minerados, 470 passos de seleção de regra aplicável foram executados, ou seja, 470 vezes o motor de inferência foi chamado. Ainda é apresentado o tamanho médio das listas de sugestões geradas durante a simulação, bem como o seu desvio padrão. O tamanho médio foi em torno de 22,74 com um desvio padrão de 20,13. O desvio padrão grande indica que o tamanho das listas de sugestões varia bastante. Ambos os números são consideráveis, sendo importante salientar que a lista de sugestões construída durante a avaliação automática possui sempre todas as regras possíveis, o que eleva estes valores. Em um caso de uso real, isto é, por um usuário humano, a lista de sugestões deve ter um limite de regras a serem apresentadas ao usuário e, por este motivo, as informações subsequentes na Figura 6.13 apresentam valores apenas para as onze primeiras posições da lista e a última para que se possa ter uma ideia de até onde é possível que regras interessantes sejam encontradas.

Seguindo a análise dos resultados exibidos na Figura 6.13 tem-se a quantidade de regras de cada posição da lista de sugestões que seriam utilizadas para dar continuidade nos desenhos dos modelos de processo simulados. Pode-se ver que, como esperado, a maior taxa de acertos concentra-se nas primeiras posições da lista. Em 11,03% das vezes, ou seja, em 52 casos, a regra apresentada na primeira posição da lista de sugestões foi a regra utilizada para dar continuidade ao desenho do modelo conforme desejado. Em 22 casos a regra da segunda posição da lista seria a utilizada. O mesmo valor surge para a terceira posição também. Acumulando-se os resultados das cinco primeiras posições da lista, o que é um valor bastante aceitável para um usuário analisar, tem-se uma taxa de acertos de aproximadamente 26,38%, ou seja, 124 casos. Acumulando-se os valores das dez primeiras posições tem-se 37,64% de sucesso em predizer as próximas estruturas a serem inseridas nos modelos simulados, ou seja, em 177 dos 470 passos executados.

```
Summary of tests:
Total number of rules on the database: 48850
Total number of steps of rule selection: 470
Average size of suggestion list: 22.74122807017544
Standard deviation of suggestion list: 20.133372050823503

Suggestion list results:
Rules matched on position 1: 52 11.063829787234043%
Rules matched on position 2: 22 4.680851063829787%
Rules matched on position 3: 22 4.680851063829787%
Rules matched on position 4: 14 2.978723404255319%
Rules matched on position 5: 14 2.978723404255319%
Rules matched on position 6: 12 2.5531914893617023%
Rules matched on position 7: 20 4.25531914893617%
Rules matched on position 8: 7 1.4893617021276595%
Rules matched on position 9: 5 1.0638297872340425%
Rules matched on position 10: 9 1.9148936170212767%
Rules matched on position 11: 5 1.0638297872340425%
…
Rules matched on position 45: 1 0.2127659574468085%
Unmatched rules: 250 53.191489361702125%

Rules level count results:
Rules matched of level 1: 169 35.95744680851064%
Rules matched of level 2: 35 7.446808510638298%
Rules matched of level 3: 6 1.2765957446808511%
Rules matched of level 4: 4 0.851063829787234%
Rules matched of level 5: 2 0.425531914893617%
Rules matched of level 6: 2 0.425531914893617%
Rules matched of level 7: 2 0.425531914893617%
```

Figura 6.13: Resultado de testes com conjunto de treinamento e validação separados

Pode-se verificar que ocorreu um acerto da lista de sugestões com uma regra que estava na 45ª posição desta. Em 53,19% dos casos a lista de sugestões não contemplou o desejo do usuário, o que quer dizer que a regra que deveria descrever a evolução do modelo sendo desenhado não estava presente.

A próxima seção de resultados da Figura 6.13 apresenta a quantidade de regras utilizadas segmentadas por tamanho. Por tamanho considera-se o número de arestas do antecedente da regras. Desta maneira, regras de nível 1 possuem como antecedente um modelo com uma aresta, regras de nível 2 possuem como antecedente um modelo de 2 arestas e assim sucessivamente. Pode-se ver que a maior parte das regras utilizadas foram as de tamanho um, dois e três, correspondendo a mais de 44,5% dos casos.

Após a realização desta avaliação por particionamento estratificado aleatório, foi realizado um teste utilizando-se todos os modelos de processo disponíveis para minerar os padrões e ter a sua construção simulada com o auxílio destes. Os resultados são apresentados na Figura 6.14.

```
Summary of tests:
Total number of rules on the database: 49530
Total number of steps of rule selection: 1425
Average size of suggestion list: 49.107877927608236
Standard deviation of suggestion list: 183.19469198715663

Suggestion list results:
Rules matched on position 1: 259 18.17543859649123%
Rules matched on position 2: 132 9.263157894736842%
Rules matched on position 3: 102 7.157894736842105%
Rules matched on position 4: 57 4.0%
Rules matched on position 5: 45 3.1578947368421053%
Rules matched on position 6: 33 2.3157894736842106%
Rules matched on position 7: 25 1.7543859649122806%
Rules matched on position 8: 16 1.1228070175438596%
Rules matched on position 9: 27 1.894736842105263%
Rules matched on position 10: 17 1.1929824561403508%
Rules matched on position 11: 24 1.6842105263157894%
…
Rules matched on position 79: 1 0.07017543859649122%
Unmatched rules: 528 37.05263157894737%

Rules level count results:
Rules matched of level 1: 486 34.10526315789474%
Rules matched of level 2: 259 18.17543859649123%
Rules matched of level 3: 74 5.192982456140351%
Rules matched of level 4: 31 2.175438596491228%
Rules matched of level 5: 19 1.3333333333333333%
Rules matched of level 6: 13 0.9122807017543859%
Rules matched of level 7: 9 0.631578947368421%
Rules matched of level 8: 2 0.14035087719298245%
Rules matched of level 9: 2 0.14035087719298245%
Rules matched of level 10: 2 0.14035087719298245%
```

Figura 6.14: Resultado de testes com todos os modelos

Neste caso, foram geradas 49530 regras a partir dos 8178 padrões frequentes encontrados anteriormente, conforme apresentado no Capítulo 5. Nesta simulação, foram executados 1425 passos de cálculo da lista de sugestões de padrões e escolha do padrão desejado caso este se apresente na lista. A média de tamanho da lista de sugestões foi em torno de 49 regras, com um desvio padrão de mais de 183. Da mesma maneira que o teste anterior, estes valores grandes indicam que o tamanho das listas de sugestões varia bastante ao longo das simulações.

Analisando-se a quantidade de regras de cada posição da lista de sugestões que se apresentaram corretamente para a continuidade da construção do modelo, tem-se que a maior taxa de acertos está no topo da lista com mais de 18% dos casos, ou seja, em 259 vezes. A segunda posição da lista apresentou regras úteis em 132 casos, ou 9,26% do total. Considerando-se as cinco primeiras posições da lista de sugestões a taxa de acerto fica em 41,75%, 595 casos. Indo mais além, nota-se que as dez primeiras posições da lista possuem uma regra que indique corretamente a próxima estrutura a ser inserida no modelo em 50% das vezes, ou 713 casos.

A saída dos testes ainda demonstra que se chegou a analisar uma lista de sugestões de tamanho 79 ou maior, já que foi encontrada uma regra útil nesta posição da lista de sugestões. 528 passos de modelagem não estavam presentes nas regras computadas, o que representa aproximadamente 37% do total de passos simulados.

O último grupo de informações da Figura 6.14 exibe a quantidade de regras corretamente sugeridas conforme o nível de cada uma delas. O que se pode observar é que a maior parte dos acertos concentram-se nas regras de nível 1 a 3 novamente. Conforme será visto na Seção 6.4, esta informação pode ser utilizada para filtrar as regras da lista de sugestões a serem apresentadas. Com isto, pode-se possivelmente melhorar a quantidade de acertos entre as primeiras posições.

Para efeitos de comparação, a chance de se acertar a próxima aresta/nodo a serem inseridos em um modelo de processo, considerando os 16 tipos de nodos apresentados no Capítulo 5 e descontando-se o nó de início, é de 6,25%. É importante observar que este cálculo é apenas para se decidir qual a próxima aresta/nodo a ser inserido. Ainda é necessário calcular onde este próximo item estará conectado. Em um modelo com dois nodos esta probabilidade cai para a metade, ou seja, 3,125%. Um modelo com três nodos permite uma chance de 2,08% de acerto. A média de tamanho dos modelos de processos analisados é de 11 nodos. Levando-se em conta metade deste tamanho, 5 nodos, a probabilidade de se acertar como o modelo deve evoluir é de 1,25%. Isto mostra como a taxa de acertos de mais de 18% na primeira posição da lista de sugestões conforme apresentado na Figura 6.14, ou de mais de 11% conforme a Figura 6.13, é importante. Pode-se verificar que o motor de inferência realmente beneficia-se dos padrões minerados para prever como o desenho dos modelos de processos pode evoluir e que ele é capaz de fazer boas sugestões para esta evolução.

A Tabela 6.1 apresenta uma comparação realizada com a utilização da ferramenta desenvolvida para verificar como o tipo de contagem dos suportes dos padrões pode influenciar nos resultados. A tabela revela as taxas de acertos das três primeiras posições da lista de sugestões conforme cada um dos dois tipos de contagem de suporte. O primeiro considera processos, ou seja, se um padrão frequente aparece em um processo uma ou mais vezes, seu suporte é contado apenas uma vez. O segundo tipo de contagem é global e conta cada ocorrência de um padrão frequente, inclusive dentro de um mesmo processo caso ele apareça mais de uma vez.

Tabela 6.1: Comparação de testes considerando dois tipos de contagens de suporte dos padrões

Posição na lista	Contagem por processo	Contagem global
1ª	17.403508771929825%	18.17543859649123%
2ª	8.350877192982455%	9.263157894736842%
3ª	5.894736842105263%	7.157894736842105%

Como se pode ver, há uma melhora na taxa de acertos nas primeiras posições da lista de sugestões quando a contagem de suporte global é utilizada. Isto se deve ao fato de que a contagem global representa melhor a realidade. A ocorrência ou não das associações entre os padrões de atividade não é dependente de outras ocorrências no mesmo processo. Por este motivo, a contagem do suporte por processos não captura toda a realidade. Contando-se todas as associações frequentes dos padrões de atividade e utilizando-as para prever comportamentos futuros os resultados são melhores, pois regras que possuem associações recorrentes que aparecem mais vezes em um mesmo processo tendem a aproximar-se das primeiras posições da lista de sugestões. Como estas regras são realmente frequentes, as primeiras posições da lista de sugestões passam a apresentar alternativas com maior probabilidade de serem corretas e, por isto, a maior taxa de acertos.

6.4 Filtragem de Regras

Conforme pôde ser visto na Seção 6.3, existe uma certa predominância das regras de tamanho 1 a 3 nos resultados dos testes executados. Em vista desta observação, a seguir são apresentados estudos onde foram implementados filtros de regras geradas. Estes filtros eliminam regras que não se deseja que sejam avaliadas e influenciam na lista de sugestões gerada pelo motor de inferência.

Foram testadas cinco configurações de filtros diferentes: uma gerando apenas regras de tamanho 1; uma gerando regras de tamanho 1 e 2; outra gerando regras de tamanho 1 a 3; regras de tamanho 1 a 4; e a última gerando regras de tamanho 1 a 5.

A Tabela 6.2 apresenta algumas informações resultantes deste estudo e permite analisar como o número de níveis considerados no modelo antecedente, isto é, o tamanho das regras, influencia na qualidade da lista de sugestões gerada.

Tabela 6.2: Comparação de diferentes implementações de filtros

Tamanho das regras:	1	1 e 2	1 a 3	1 a 4	1 a 5	Todas
Regras geradas	414	1109	1757	2941	5626	49530
Acertos na 1ª posição	13,45%	16,21%	17,40%	17,68%	17,40%	18,17%
Acertos na 2ª posição	8,42%	8,77%	9,26%	9,47%	9,12%	9,26%
Acertos na 3ª posição	4,35%	7,15%	6,24%	6,31%	7,01%	7,15%
Passos não previstos	42,80%	38,10%	37,75%	37,68%	37,19%	37,05%

Em uma análise sobre estes números pode-se ver que a filtragem de regras reduz drasticamente o número total de regras geradas e que precisam ser analisadas. Isto possibilita otimização no tempo de execução dos algoritmos, já que menos processamento é requerido. Uma outra observação que pode ser feita é que a utilização

de regras apenas de tamanho 1 tem os piores resultados. Tal fato evidencia a importância de se considerar não somente o último padrão inserido em um modelo para predizer o próximo, mas a necessidade de se levar em conta o modelo como um todo, ou pelo menos, os últimos padrões inseridos. Outro fato interessante a ser observado nos números da Tabela 6.2 está na comparação dos resultados da utilização de filtros de regras de tamanho 1 a 4 e filtros de regras de tamanho 1 a 5. Nesta mudança pode-se notar que existe uma queda na taxa de acertos na primeira e segunda posição da lista de sugestões e um aumento da taxa de acertos na terceira posição. Isto acontece, pois no segundo caso existem mais regras. Estas regras acabam por serem inseridas na lista de sugestões em alguns casos, mas são ineficientes, pois a regra que prediz o comportamento correto é uma regra que já estava presente no primeiro caso. Quando existem regras adicionais, as regras que apareciam anteriormente tendem a aparecer em outras posições da lista de sugestões mais distantes do topo, já que esta é ordenada pelo tamanho das mesmas. Por este motivo, os acertos nas duas primeiras posições diminuem e os acertos na terceira posição aumentam. Isto quer dizer que as regras que predizem a evolução desejada do modelo estavam aparecendo antes nas primeiras posições da lista e agora passaram a aparecer na terceira. A Figura 6.15 apresenta a informação da Tabela 6.2 em um gráfico para que a evolução dos números possa ser vista.

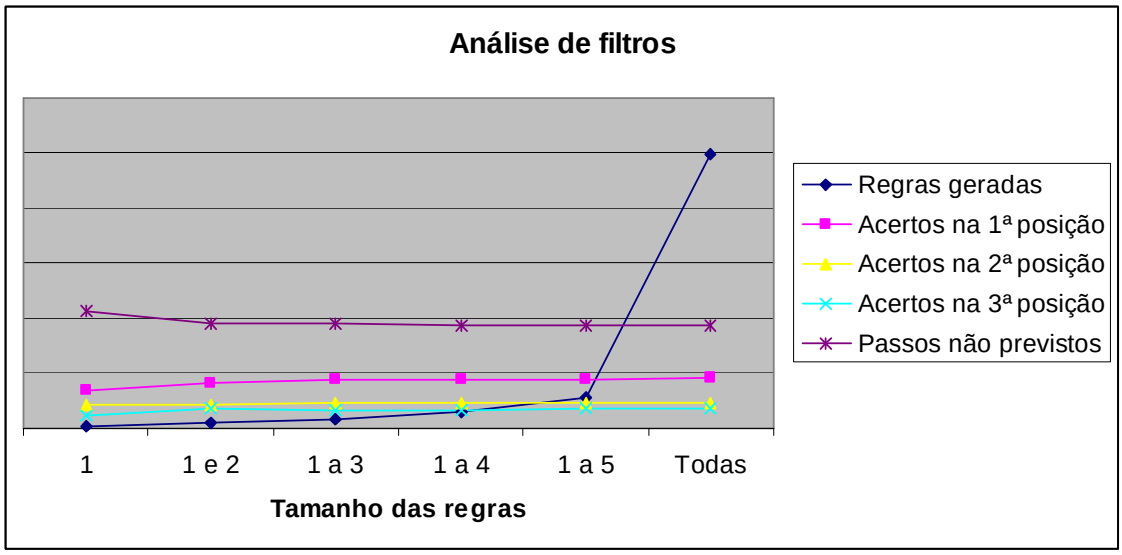

Figura 6.15: Gráfico comparativo de diferentes filtros

A análise do gráfico da Figura 6.15 permite verificar que há uma explosão na quantidade de regras geradas conforme os filtros tornam-se menos restritivos. Quanto maiores os modelos que podem ser antecedentes de regras geradas, maiores as possibilidades de regras diferentes serem geradas e, por isto, esta explosão.

Outro fato a ser observado é que as curvas das taxas de acertos e de erros possuem certa melhoria até o momento em que regras de tamanho 1 a 3 são consideradas. A partir deste ponto, o gráfico indica que a consideração de mais regras traz pequenos benefícios, mas que podem não ser interessantes. A geração de mais regras pode ter vantagens por trazer regras mais específicas. Porém, estas regras podem não ser úteis na maioria dos casos e podem prejudicar a taxa de acertos da lista de sugestões entre as primeiras posições. Deste modo, é importante que seja feito um balanceamento entre esses ganhos e perdas para que regras inúteis não sejam geradas e tempo de processamento não seja desperdiçado.

Com isto, pode-se dizer que a utilização de regras de até três níveis, ou seja, contendo no máximo quatro padrões de atividade em seu antecedente, é razoável para uma boa predição da próxima estrutura a ser adicionada aos modelos parciais, sem comprometer os resultados e o desempenho dos algoritmos. Mais do que isto faz com que mais regras sejam computadas, porém com pouco benefício.

6.5 Analisando Categorias de Processos

Aqui é apresentado um último estudo que foi realizado com a utilização da ferramenta desenvolvida. Este estudo contempla a execução da ferramenta sobre dois grupos de processos. Estes grupos representam categorias e o objetivo desta investigação é verificar se alguma categorização dos processos é capaz de trazer melhores resultados de predição quando comparados aos resultados onde os processos não são categorizados. Para este estudo, o atributo selecionado para classificar os processos refere-se ao tipo das atividades que são executadas. Neste contexto, os diferentes tipos considerados são dois: automatizadas e manuais. Assim, os processos foram divididos em dois grupos: um contendo processos cuja maioria das atividades são automatizadas e outro grupo contendo processos cujas atividades são manuais em sua maioria. Cada um dos grupos foi separadamente submetido à mineração de dados para a identificação dos padrões recorrentes em cada um deles. Em seguida, cada conjunto de padrões identificados foi submetido à avaliação automatizada da ferramenta para verificar se o conhecimento minerado em uma categoria específica de processos pode trazer vantagens nos resultados obtidos.

A Tabela 6.3 apresenta os resultados verificados utilizando-se os padrões minerados sobre o conjunto de processos cuja maioria das atividades são manuais e a sua validação utilizando-se este mesmo conjunto, bem como o conjunto de processos de maioria de atividades automatizadas para a realização de uma comparação. A última coluna apresenta os resultados da mineração e validação executadas utilizando-se todos os processos, conforme já apresentado na Seção 6.3.

Tabela 6.3: Resultados da mineração de processos cuja maioria das atividades são manuais

	Processos manuais	Processos automatizados	Todos os processos
Passos simulados	1184	241	1425
Regras geradas	49108	49108	49530
Acertos na 1ª posição	15,12%	14,94%	18,16%
Acertos na 2ª posição	9,54%	3,32%	9,26%
Acertos na 3ª posição	5,74%	2,90%	7,16%
Acertos até a 10ª posição	46,85%	40,65%	50%
Passos não previstos	40,71%	43,15%	37,05%

Um primeiro fato a ser observado é que o conjunto de processos automatizados ficou menor do que o conjunto de processos manuais. Isto é notado ao observar-se a linha da Tabela 6.3 onde está registrado o número de passos simulados. No conjunto de processos manuais foram simulados 1184 passos, enquanto que no outro conjunto de processos foram simulados 241 passos.

A quantidade de regras geradas é a mesma quando utilizado tanto o conjunto de processos manuais como o conjunto de processos automatizados para a avaliação, pois as regras foram geradas a partir dos padrões frequentes minerados sobre o primeiro conjunto. Como contraponto, a última coluna apresenta a quantidade de regras geradas quando foram utilizados todos os processos para a mineração de dados. O que pode ser observado é que o número de regras diminui levemente. Possivelmente isto se deve ao fato de que a maioria dos processos utilizados neste estudo pertence à categoria de processos manuais. Um outro fator que pode ter contribuído para esta pequena redução no número de regras geradas é que os processos manuais são bastante diversificados. Em virtude desta diversidade, uma grande quantidade de padrões frequentes diferentes acaba sendo encontrada neste conjunto, o que acarreta na grande quantidade de regras geradas.

Comparando-se a quantidade de acertos da lista de sugestões quando avaliados os processos manuais e quando avaliados os processos automatizados pode-se notar que há uma maior taxa de acertos para os processos do primeiro grupo. O inverso acontece para a porcentagem de passos onde a lista de sugestões não previu a evolução da modelagem. Este resultado é o esperado, visto que os processos manuais devem ter uma taxa de predição melhor quando são utilizados os padrões frequentes minerados neste mesmo conjunto de processos. No entanto, as diferenças entre as taxas de acertos nos dois conjuntos não foram muito grandes. Isto indica que os padrões frequentes encontrados nos processos manuais podem ser encontrados em uma proporção semelhante nos processos automatizados. Em outras palavras, pode-se dizer que os padrões encontrados nos processos manuais são, em sua maioria, padrões comuns a modelos de processos em geral, que podem também ser encontrados em processos mais automatizados.

Comparando-se as taxas de acertos e erros da validação utilizando os processos manuais com as taxas de acertos e erros do estudo realizado utilizando-se todos os processos, pode-se verificar que há uma leve queda na taxa de acertos e um pequeno aumento na taxa de não previsões. Acredita-se que isto se deve ao fato de que os padrões que seriam encontrados na maioria dos processos automatizados e não foram encontrados nos processos manuais tenham causado este pequeno aumento na taxa de não previsões. Isto indica que existe uma pequena parcela de processos mais manuais que possui certos padrões recorrentes que aparecem com mais frequência nos processos mais automatizados.

Pode ser observado também que a taxa de acertos na primeira posição da lista teve um decréscimo, enquanto que a taxa de acertos na segunda posição aumentou. Uma explicação para este acontecimento é que com a utilização apenas de processos manuais o suporte e confiança das regras foi alterado. Assim, as regras que aparecem tanto em processos manuais como em processos automatizados podem ter seu suporte diminuído se considerarmos apenas processos manuais. Em contrapartida, as regras que aparecem em sua maioria apenas nos processos manuais acabam tendo o seu suporte aumentado. Isto faz com que a ordem de apresentação na lista seja modificada. Esta mudança pode fazer com que nos casos em que uma regra que prediz como o modelo deve evoluir que antes aparecia na primeira posição da lista passe a ser exibida na segunda posição, principalmente se esta regra for comum a processos dos dois tipos. Neste caso, uma regra característica de processos manuais pode assumir a primeira posição da lista e fazer com que a regra que prediz a evolução correta em alguns casos caia algumas posições.

A Tabela 6.4 apresenta os resultados para o mesmo experimento, porém utilizando-se os padrões minerados sobre o conjunto de processos mais automatizados. O número de passos simulados é o mesmo do experimento da Tabela 6.3, pois os mesmos processos foram utilizados. A quantidade de regras geradas neste caso foi drasticamente menor do que o caso anterior e se comparada com a quantidade de regras geradas com a utilização de todos os processos. Isto acontece porque o conjunto de padrões frequentes resultante da etapa de mineração de dados é menor, o que indica que os processos mais automatizados são mais homogêneos.

Tabela 6.4: Resultados da mineração de processos cuja maioria das atividades são automatizadas

	Processos manuais	Processos automatizados	Todos os processos
Passos simulados	1184	241	1425
Regras geradas	216	216	49530
Acertos na 1ª posição	2,45%	35,68%	18,16%
Acertos na 2ª posição	1,86%	13,28%	9,26%
Acertos na 3ª posição	1,94%	3,32%	7,16%
Acertos até a 10ª posição	8,84%	54,35%	50%
Passos não previstos	90,79%	45,23%	37,05%

Comparando-se as taxas de acertos nas primeiras posições da lista de sugestões dos processos automatizados com os resultados obtidos utilizando-se todos os processos pode-se notar uma nítida melhoria. Esta observação novamente confirma que os processos mais automatizados são mais homogêneos, o que torna a sua predição mais fácil de ser realizada e com um menor número de regras. Por este motivo, praticamente todos os acertos da lista de sugestões no caso dos processos automatizados concentram-se nas três primeiras posições. O número de passos não previstos teve um incremento pelo mesmo motivo daquele verificado no estudo anterior: existem algumas recorrências que são comuns aos dois grupos de processos as quais deixam de ser capturadas quando eles são estudados em separado.

Verificando os números da validação sobre os processos manuais encontra-se um resultado bastante interessante. As taxas de acertos são mínimas e a taxa de passos não previstos dispara. Isto confirma a hipótese de que existem recorrências que são comuns a determinados tipos de processos apenas. Neste caso, o desenho dos processos manuais foi tentado ser previsto utilizando-se os padrões minerados sobre os processos automatizados. Pode-se ver que a tentativa apresentou resultados muito ruins e a conclusão é que os padrões minerados sobre os processos mais automatizados não se aplicam aos outros tipos de processos. A alta taxa de passos não previstos revela que os padrões minerados são praticamente exclusivos de processos mais automatizados.

Através deste estudo é evidenciada a ideia de que a consideração de características dos processos para a mineração de dados pode trazer resultados interessantes e melhorar a taxa de acertos em alguns casos. Existem certos padrões que são mais comuns a processos de determinado tipo, ficando a tarefa de predição mais otimizada quando estas características podem ser levadas em conta.

7 CONCLUSÃO

A incorporação da tecnologia de *workflow* ao cotidiano das organizações atuais é fator decisivo para que estas continuem competitivas e desempenhem seus serviços da melhor maneira possível. Sabendo desta necessidade, as companhias têm trabalhado nesta direção e a modelagem de processos de negócio tem sido bastante pesquisada ao longo dos últimos anos.

Conforme visto, na etapa de modelagem de processos determinadas estruturas recorrentes têm sido identificadas. Estas estruturas são classificadas como padrões e têm de ser redesenhadas a cada nova modelagem. Neste sentido, este trabalho procurou desenvolver uma técnica ou método que permita descobrir e analisar associações entre padrões de atividade de modelos de processo. A aplicação desta técnica permitiu obter informação suficiente para o desenvolvimento de um "motor de inferência" capaz de auxiliar o usuário durante a etapa de modelagem de processos com base em padrões de atividades.

Os resultados deste trabalho de pesquisa podem facilitar a modelagem baseada na utilização de estruturas recorrentes, o que padroniza a construção dos modelos e contribui para que os processos tornem-se menos suscetíveis a erros e inconsistências. Além disso, o desempenho e a qualidade da etapa de modelagem são beneficiados quando padrões são utilizados. O uso de padrões orientados à organização, na modelagem de processos, pode melhorar a qualidade de um modelo na medida em que ele melhor representará os processos de negócio reais como eles são realizados pela organização. Adicionalmente, com relação aos padrões, se estes são suportados por uma ferramenta de modelagem, eles resultam em *templates* de atividades orientados a objetivos. O projetista, após selecionar um padrão, só necessita completar o *template* com os atributos específicos daquele trecho de seu processo. Comparado ao tempo de construir o trecho completo, do zero, o tempo de selecionar o *template* e *customizá-lo* pode ser bem menor. Se isso ficar provado, em trabalhos futuros, o uso de padrões na modelagem dos processos trará aumento de produtividade além de melhorar a qualidade e a documentação dos modelos.

A base da metodologia adotada para a obtenção das informações desejadas foi buscada no campo da Inteligência Artificial, mais precisamente, na Descoberta de Conhecimento em Bases de Dados. Durante a aplicação desta metodologia foi observada a dificuldade em lidar com dados e conhecimento que mudam. Isto é, sempre que a base de dados alvo do processo de descoberta de conhecimento é alterada, o resultado do processo precisa ser atualizado. Para que isto seja feito, muitas vezes é

necessária a re-execução de todas as etapas do processo. Em virtude disto, ficou evidenciada a forte necessidade do desenvolvimento de ferramentas computacionais auxiliares capazes de automatizar algumas etapas e permitir que toda a massa de dados possa contribuir para a geração dos melhores resultados e tornar a sua análise factível.

Ao longo dos estudos sobre os diferentes objetivos e técnicas de descoberta de conhecimento foi verificada a vasta gama de áreas onde esta metodologia pode ser aplicada. Uma grande quantidade de problemas podem ser abordados sob esta perspectiva e diversas análises podem ser realizadas. O próprio algoritmo de mineração de dados utilizado neste trabalho é genérico o suficiente a ponto de poder ser aplicado a inúmeros outros domínios diferentes daquele de modelos de processos. Em vista disto, é muito importante que os resultados esperados estejam bem caracterizados a fim de que se possa definir qual a melhor estratégia a ser seguida para o sucesso do estudo pretendido.

A abordagem seguida neste trabalho pode ser considerada inovadora no estudo de modelos de processos, já que a maior parte dos trabalhos sobre mineração de processos é baseada em *logs* de execução dos processos e não na mineração de modelos de processos. Na mineração de *logs*, os processos precisam ser modelados, implementados e executados muitas vezes para padrões de execução serem identificados. A vantagem da mineração de modelos é que as informações podem ser extraídas tão logo o processo esteja modelado. Não é necessária nem a implementação do processo executável, nem sua execução repetida.

Um outro ponto a ser observado quanto à etapa de mineração, é o fato de que ela é bastante configurável. Por exemplo, podem ser especificados valores de suporte mínimo e se devem ser gerados apenas padrões máximos ou todos os padrões. Variações destes parâmetros produzem resultados diferentes, que podem ser analisados de diferentes formas, conforme os objetivos desejados e as técnicas disponíveis para tal.

Os resultados atingidos através do estudo de caso com aplicação da técnica proposta foram significativos para o estudo dos padrões de atividade, suas associações recorrentes e respectiva disponibilização durante a fase de modelagem. As informações obtidas confirmam a hipótese inicial da alta frequência no reuso de estruturas, em processos, e apontam para o acerto na escolha da metodologia de estudo. Uma indicação disto foi obtida quando os padrões máximos foram verificados e foi encontrado um padrão de tamanho 8 com alto suporte. Ao verificar-se onde este padrão era encontrado, foi constatado que todos eram sub-processos de envio e recebimento de algum objeto. Isto demonstra o poder do algoritmo de mineração que através da mineração somente de modelos, foi capaz de encontrar este mesmo padrão em 7 processos.

Ao longo do desenvolvimento da ferramenta de pós-processamento foi notada a importância do bom uso da orientação a objetos e correta modularização do código. Tais características possibilitaram a implementação de um código facilmente manutenível e expansível, de modo que novas funcionalidades possam ser adicionadas sem maiores complicações e sem risco de, inadvertidamente, afetar-se outras porções do código.

Os resultados obtidos a partir da utilização do BPM Miner foram bastante interessantes e mostraram que a sugestão automática de padrões pode chegar a taxas de acertos interessantes. Comparando-se com resultados onde nenhuma informação *a priori* é utilizada, foi possível ver como as informações obtidas na etapa de mineração de dados são importantes e descrevem associações características que aparecem comumente em modelos de processos.

Através da implementação de filtragem nas regras geradas pela ferramenta de pós-processamento, verificou-se a importância de considerar não somente o último padrão inserido, no modelo, para sugerir o próximo, mas que um bom nível de acerto já é atingido quando pelo menos os quatro últimos padrões já modelados são considerados na predição de um próximo padrão. Este resultado confirma a hipótese inicial de que a consideração do modelo, como um todo, na predição de um próximo padrão pode trazer melhores resultados em relação à consideração de somente o último padrão inserido.

Outro resultado que confirmou suspeitas iniciais foi aquele obtido no experimento utilizando a ferramenta com os processos segmentados conforme suas características. Verificou-se que a consideração de características dos processos podem ser interessantes de serem levadas em conta durante a mineração. O resultado obtido evidenciou a existência de padrões específicos aos processos mais automatizados e traz motivação para que segmentações de processos conforme outras características sejam estudadas, em trabalhos futuros, na busca de padrões específicos em determinados tipos de processos como, por exemplo, a partir da segmentação dos processos em domínios de aplicação distintos.

Como trabalhos futuros diretamente derivados da presente pesquisa tem-se, em primeiro lugar, a realização de novos e diferentes estudos de caso com o auxílio da ferramenta desenvolvida. Um número maior de modelos pode ser incorporado à base de informações, estudos de como outras segmentações por características dos processos podem afetar os resultados e estudos com a utilização de desenhistas reais de modelos de processo auxiliados pelas regras sugeridas pelo motor de inferência desenvolvido podem ser realizados. O estudo com projetistas reais pode, inclusive, utilizar a sugestão automatizada de partes maiores de processos em vez de sugerir somente o próximo padrão de atividade que deve ser inserido em um modelo parcial. Desta maneira, ao sugerir seqüências mais complexas de padrões de atividade conectados, pode-se verificar se o ganho de produtividade e desempenho desta fase pode ser maior.

Um segundo ponto onde ainda existe trabalho a fazer é a ampliação e modificação da ferramenta. Diferentes filtros de regras podem ser implementados e testados, assim como diferentes algoritmos de ordenação das regras da lista de sugestões podem ser analisados.

Um aspecto interessante a ser explorado nesta ampliação da ferramenta é a atualização da lista ordenada de sugestões, de forma automática, com base nas escolhas que os usuários vão fazendo a partir dos padrões a eles sugeridos em cada situação. Isto é, avaliar quais têm sido as sugestões mais utilizadas da lista e promovê-las para as primeiras posições, conforme a sua seleção tornar-se mais frequente. Uma maneira direta para realizar isto é adicionar um contador para armazenar o número de vezes que uma sugestão foi escolhida pelo usuário e levar este valor em conta para a ordenação da lista.

Outro trabalho interessante a ser desenvolvido envolve a automação de partes que ainda foram manualmente realizadas nesta pesquisa. Uma destas etapas é a identificação dos padrões de atividade nos modelos de processos. Outra etapa interessante de ser automatizada seria a segmentação dos processos conforme suas características para posterior análise de associações recorrentes.

A seguir, estão as referências a dois artigos relacionados com o trabalho desenvolvido no âmbito desta dissertação e que foram publicados no ICEIS

(*International Conference on Enterprise Information Systems*), Qualis B2 Internacional, em 2007 e 2009, respectivamente:

THOM, L. H.; LAU, J. M.; IOCHPE, C.; MENDLING, J. Extending Business Process Modeling Tools With Workflow Pattern Reuse. Em**: International Conference on Enterprise Information Systems, ICEIS**, Funchal, Portugal, v.9, 2007.

LAU, J. M.; IOCHPE, C.; THOM, L. H.; REICHERT, M. Discovery and Analysis of Activity Pattern Co-occurrences in Business Process Models. Em**: International Conference on Enterprise Information Systems, ICEIS**, Itália, Milão, v.11, 2009.

REFERÊNCIAS

AALST, W. M. P. van der et al. Workflow Patterns. **Distributed and Parallel Databases**, Eindhoven, v.14, p. 5-51., 2003.

AALST, W. M. P. van der. YAWL: Yet Another Workflow Language. **Information Syst.**, p. 245-275, 2005.

AALST, W. M. P. van der. Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing. In: **Requirements Engineering Journal**, 10(3), p.198-211, 2005.

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A.. Mining Association Rules between Sets of Items in Large Databases. In: **Proceedings, ACM SIG-MOD Conference on Management of Data**, Washington, D.C., p.207-216, 1993.

AGRAWAL, R. et al.. Fast Discovery of Association Rules, American Association for Artificial Intelligence, In: **Advances in Knowledge Discovery and Data Mining**, Menlo Park, CA, p. 307-328, 1996.

AIBER, S. et al. **Business Objective Based Resource Management**. In: 13th International World Wide Web Conference, **Proceedings…**, Haifa, p. 236-237, 2004.

ALLEN, R.. Workflow: An Introduction. In: **Workflow HandBook**, Reino Unido, p. 15-38, 2001.

AVERSON, P. **The Deming Cycle**. Balanced Scorecard Institute, 1998. Disponível em: <http://www.balancedscorecard.org/>. Acesso em: Maio 2006.

BECKER, J.; PFEIFFER, D.; RÄCKERS, M.. Domain Specific Process Modelling in Public Administrations - The PICTURE Approach. In: **Proc. EGOV'07**, p. 68-79, 2007.

BRADSHAW, D.; KENNEDY, M.; WEST, C. **Oracle BPEL Process Manager. Developer's Guide**. Release 2 (10.1.2). 2005. Disponível em: <http://www.oracle.org>. Acesso em: Outubro 2006.

CASATI, F., et al. Business Process Intelligence. **Computers in Industry**, 53(3), pp. 321-343, 2004.

CHAKRAVARTHY, S.; BEERA, R.; BALACHANDRAN, R.. DB-Subdue: Database Approach to Graph Mining. In: **Lecture Notes in Computer Science**, Springer Berlin, v. 3056, p.341-350, 2004.

CHAKRAVARTHY, S.; PRADHAN, S.. DB-FSG: Na SQL-Based Approach for Frequent Subgraph Mining. In: **Lecture Notes in Computer Science**, Springer Berlin, v. 5181, p. 684-692, 2008.

CHEN, Z. Computational Intelligence for Decision Support, CRC Press, Part III - **Chapter 10: From Machine Learning to Data Mining**, p. 225-256, 2000.

CUNTZ, N. et al. **On the semantics of EPCs: Efficient calculation and simulation**. Luxemburgo. 2004. Disponível em: <http://wwwcs.uni-paderborn.de>. Acesso em: Outubro 2006.

ELLIS, C. Workflow Mining: Definitions, Techniques, Future Directions. **2006 Workflow Handbook**, p.213-228, 2006.

FAYYAD, U.; SHAPIRO-PIATETSKY, G.; SMYTH, P. From Data Mining to Knowledge Discovery in Databases. **AI Magazine**, [s.l.], v.17, n.3, p. 37-54, 1996.

FISCHER, I,; MEINL, T.. Graph Based Molecular Data Mining – An Overview. In: **2004 IEEE International Conference on Systems**, Man and Cybernetics, v.5, p.4578-4582, 2004.

FUNDAÇÃO NACIONAL DA QUALIDADE. **Critérios de Excelência**. O estado da arte da gestão para a excelência do desempenho e para o aumento da competitividade. 2006. Disponível em <http://www.fnq.org.br/Portals/_FNQ/Documents/CE2006BR_Rev1.pdf>. Acesso em: Outubro, 2007.

GOEBEL, M.; GRUENWALD, L.. A survey of data mining and knowledge discovery software tools, In: **SIGKDD Explorations**, v.1, 1999. Disponível em: <http://citeseer.ist.psu.edu/goebel99survey.html>. Acesso: Maio, 2008.

GÜNTHER, C. W. et al. Using Process Mining to Learn from Process Changes in Evolutionary Systems. In: **Int. Journal of Business Process Integration and Management**, v.3(1), p.61-78, 2008.

HAN, J.; PEI, J.; YIN, Y. Mining Frequent Patterns without Candidate Generation. **ACM SIGMOD Intl. Conference on Management of Data**, [s.l.], 2000.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**, Morgan Kaufmann, 550 p., San Diego, CA, 2001.

INTALIO. **Creating Process Flows**. 2006. [s.l.]. Disponível em: <http://bpms.intalio.com>. Acesso em: Outubro, 2007.

IOCHPE, C.; THOM, L. H.. Relying on the Organizational Structure to Model Workflow Processes. In: International Conference on Enterprise Information Systems, ICEIS, 3, **Proceedings…**, Setúbal, v.2, p.740-744, 2001.

JGRAPH. **JGraph: visualize everything.** [S.l.:s.n], 2001. Disponível em: <http://www.jgraph.com>. Acesso em: Junho, 2008.

JIMÉNEZ, A.; BERZAL, F.; CUBERO, J. Mining Different Kinds of Trees: A Tree Mining Overview. **II Congresso Español de Informática – IV Taller de Minería de Datos y Apredizaje**, Zaragoza, p. 343-352, Setembro, 2007.

KIEPUSZEWSKI, B. **Expressiveness and Suitability of Languages for Control Flow Modeling in Workflows**. 2003. 207f. Dissertação, Queensland University of Technology, Brisbane.

KURAMOCHI, M.; KARYPIS, G. An Efficient Algorithm for Discovering Frequent Subgraphs. **IEEE Transactions on Knowledge and Data Engineering**, Minneapolis, v.16, n.9, p. 1038-1051, Setembro, 2004.

LARMAN, C. **Utilizando UML e Padrões**: uma introdução à análise e ao projeto orientados a objetos. Trad. Luiz A. Meirelles Salgado, Bookman, Porto Alegre, 2000.

LI, C.; REICHERT, M.; WOMBACHER, A. Discovering reference process models by mining process variants. In: **Proc. 6th Int'l Conference on Web Services (ICWS'08)**, Beijing, China, IEEE Computer Society Press, 2008.

LOPES, F. **Suporte à Reengenharia de Processos de Negócio com base em Sistemas de Workflow**. 2003. 74 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

MENDLING, J. NÜTTGENS, M. **EPC Markup Language (EPML)**. Technical Report JM-2005-03-10. Vienna University of Economics and Business Administration, 2005.

MUEHLEN, M. zur. **Workflow-based Process Controlling**. Foundation, Design and Application of workflow-driven Process Information Systems. Logos, Berlin. 2004.

MUTSCHLER, B., REICHERT, M., BUMILLER, J.. Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. In: **IEEE Computer Society Press**, IEEE Transactions on Systems, Man, and Cybernetics (Part C), [s.l.], v.38, n.3, p. 280-291, 2008.

NIJSSEN, S.; KOK, J. N. A Quickstart in Frequent Structure Mining can make a Difference. In: **Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining**, Seattle, EUA, p. 647-652, 2004.

OASIS. **Web Services Business Process Execution Language**. Version 2.0, [s.l.], 2006.

OBJECT MANAGEMENT GROUP. **Business Process Modeling Notation Specification**. Final Adopted Specification, [s.l], Fevereiro, 2006.

OWEN, M.; RAJ, J.. **BPMN and Business Process Management**. Introduction to the New Business Process Modeling Standard, Popkin Software, [s.l.], 2003.

PADMANABHAN, S. **HDB-Subdue**, A Relational Database Approach to Graph Mining and Hierarchical Reduction. 2005. 99f. Dissertação (Mestrado em Ciência e Engenharia da Computação) – University of Texas, Arlington, EUA.

PLESUMS, C. Introduction to workflow. In: **Workflow Handbook 2002**, Austin, p. 18-38, 2002.

REICHERT, M.; RINDERLE, S.; DADAM, P.. ADEPT Workflow Management System: Flexible Support for Enterprise-Wide Business Processes. In: **Lecture Notes in Computer Science**, Springer Berlin, v.2678/2003, p.1020, 2003.

RUSSELL, N.. HOFSTEDE, A. H. M ter. EDMOND, D. **Workflow Data Patterns**, [s.l.], QUT Technical report, FIT-TR-2004-01, 2004.

RUSSEL, N. et al. **Workflow Resource Patterns**. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven,. 2004.

RUSSEL, N.; AALST, W. M. P. van der; HOFSTEDE, A. Ter. Workflow Exception Patterns. In: International Conference on Advanced Systems Engineering, CAiSE, 18, 2006. **Proceedings…** [S.l.:s.n], p.288-302, 2006.

SCHOOL OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING. **Distributed Enterprise Computing – Module 1: Workflow Systems**. [S.l.], 2002.

SILVA, C. M. S. **Utilizando o Processo de Descoberta de Conhecimento em Banco de Dados para Identificar Candidatos a Padrão de Análise para Bancos de Dados Geográficos**. 2003. 146 f., Dissertação(Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, Brasil.

THOM, L. H. **Aplicando o Conhecimento Sobre os Aspectos Estruturais da Organização no Processo de Modelagem de Workflow**. (Title in English: Applying the Knowledge About Organizational Structural Aspects in Workflow Design). 2002. 120 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, Brasil.

THOM, L. H.; IOCHPE, C.; AMARAL, V. L. do; VIERO, D. M. de. Toward block activity patterns for reuse in workflow design. In: **WORKFLOW HANDBOOK 2006** including business process management : published in association with the workflow management coalition. Lighthouse Point : Future Strategies, p. 249-260, 2006.

THOM, L. H. **A Pattern Based Approach for Business Process Modeling**. 2006. 93 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

THOM, L. IOCHPE, C. Applying Block Activity Patterns in Workflow Modeling. In: **International Conference on Enterprise Information Systems (ICEIS)**, Paphos, p.457-460, 2006.

THOM, L. H.; CHIAO, C.; IOCHPE, C. Padrões de Workflow para Reuso em Modelagem de Processos de Negócio. In: **Latin American Conference in Program Languages**, SugarloafPlop, Porto de Galinhas, v.6, 2007.

THOM, L. H; LAU, J.M.; IOCHPE, C.; MENDLING, J. Extending Business Process Modeling Tools with Workflow Pattern Reuse. In: **Proceedings of 9th International Conference on Enterprise Information Systems (ICEIS 2007)**, Funchal, Madeira, Portugal, p.447-452, 2007.

THOM, L. H. et al.. Applying Activity Patterns for Developing an Intelligent Process Modeling Tool. In: **10th Int'l Conf. on Enterprise Information Systems (ICEIS'08)**, Barcelona, Spain, p. 112-119, 2008.

THOM, L. H. et al.. Inventing Less, Reusing More and Adding Intelligence to Business Process Modeling. In: **Proc. of the 19th Int'l Conference on Database and Expert Systems Applications (DEXA '08)**, Turin, LNCS 5181, p. 837-850, 2008.

THOM, L. H.; REICHERT, M., IOCHPE, C.,. Activity Patterns in Process-aware Information systems: Basic Concepts and Empirical Evidence. In: **International Journal of Process Integration and Information Management (IJPIM)**, [S.l], 2009.

TRISTÃO, C.; RUIZ, D. D.; BECKER, K.. FlowSpy: exploring Activity-Execution Patterns from Business Processes. **iSys – Revista Brasileira de Sistemas de Informação**, [s.l.], v.1, p.54-69, 2008.

WEBER, B.; RINDERLE, S.; REICHERT, M.. Change Patterns and Change Support Features in Process-Aware Information Systems. In: **Proc. 19th Int'l Conference on Advanced Information Systems Engineering (CAiSE'07)**, LNCS 4495, p. 574-588, 2007.

WEBER, B. et al.. Providing Integrated Life Cycle Support in Process-Aware Information Systems. In: **Int'l Journal of Cooperative Information Systems (IJCIS)**, 18 (1), 2009.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical machine learning tools and techniques**, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

WORKFLOW MANAGEMENT COALITION. **The Workflow Reference Model**. v1.1, TC00-1003, United Kingdom, 1995.

WORKFLOW MANAGEMENT COALITION. **Conformance White Paper**. WFMC-TC-1017, United Kingdom, 1998.

WORKFLOW MANAGEMENT COALITION. **Workflow Security Considerations – White Paper**. v1.0, WFMC-TC-1019, United Kingdom, 1998.

WORKFLOW MANAGEMENT COALITION. **Terminology & Glossary**. v3.0, WFMC-TC-1011, Bruxelas, 1999.

WORKFLOW MANAGEMENT COALITION. **Document Index**. v5.0, WFMC-TC-1002, United Kingdom, 1999.