MININGFLOW: ADICIONANDO SEMÂNTICA A WORKFLOWS DE MINERAÇÃO DE TEXTO

Daniel Cardoso Moraes de Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Profª. Fernanda Araújo Baião, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Alexandre Gonçalves Evsukoff, Dr.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2008

DE OLIVEIRA, DANIEL CARDOSO MORAES

MiningFlow: Adicionando Semântica a Workflows de Mineração de Texto [Rio de Janeiro] 2008

XV, 185 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Mineração de Texto
2. Workflows Científicos
3. Ontologias

I. COPPE/UFRJ II. Título (série)

A meus pais e meus avós.

# AGRADECIMENTOS

Provavelmente durante todo o período em que estive no mestrado eu desejei escrever os agradecimentos da minha dissertação. Talvez este seja o momento em que finalmente, depois de um período tão difícil, eu possa agradecer de verdade às pessoas importantes que me ajudaram a chegar até aqui.

Em primeiro lugar, à Professora Marta Lima de Queirós Mattoso, por me guiar pela vida acadêmica, me apresentando a conceitos ainda desconhecidos por mim como *workflows* científicos e mineração de textos, por participar da minha banca da tese, pela paciência, e acima de tudo, pelo profissionalismo com o qual me ajudou a desenvolver este trabalho.

À Professora Fernanda Araújo Baião, pela amizade, paciência, dedicação e acima de tudo pelo incentivo dado sempre que via que a "bola estava caindo". Sempre disposta a discutir temas de tese e arrumando um horário na sua agenda (sempre apertada!) para que pudéssemos nos reunir.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro a este trabalho, sem o qual, eu não conseguiria chegar até onde cheguei.

Ao Professor Geraldo Bonorino Xexéo, por participar da banca de avaliação deste trabalho e por estar sempre disponível para ajudar nos momentos em que precisei, sempre dando dicas valiosas, desde a graduação (nas matérias de MSI e outros tantos tópicos especiais que fiz) até o mestrado.

Ao Professor Alexandre Gonçalves Evsukoff, por participar da banca de avaliação deste trabalho e, principalmente por ter contribuído muito na execução do mesmo. Sua idéia inicial de acoplar nossa proposta com o que estava sendo desenvolvido no Programa de Engenharia Civil (PEC) da COPPE caiu como uma luva para nós. Além disso, sempre se mostrou disponível e colaborou sempre que pôde, seja dando idéias ou validando algo que havíamos implementado. A você Alexandre, o meu muito obrigado!

Ao Professor Mario Roberto Folhadela Benevides por ser um grande incentivador, desde os tempos em que me orientava em projeto final, para que eu fizesse o mestrado. Parte desta conquista dedico a ele.

Ao Professor Jano Moreira de Souza, chefe da linha de banco de dados, por ter me dado a oportunidade de fazer o mestrado.

À Patrícia Leal e Carolina Barreiros, secretárias da linha de banco de dados, sempre dispostas a me ajudar em tudo que precisei.

Aos alunos de mestrado e doutorado do Programa de Engenharia Civil da COPPE, sempre dispostos a ajudar, seja enviando suas implementações ou validando o que havíamos feito no *MiningFlow*.

Ao Professor José Roberto Blaschek pela oportunidade na COPPETEC e por todo o incentivo dado durante a caminhada.

Aos amigos Márcio Duran e Gustavo Pinto, que desde o primeiro dia em que comecei a trabalhar estão prontos para ajudar no que der e vier. Ambos possuem um conhecimento muito grande e ao mesmo tempo mantém uma humildade impressionante. À vocês dois, o meu muito obrigado.

Aos meus amigos da COPPE/UFRJ. Em especial, Cláudio Ferraz (sempre lá incentivando quando eu achava que as coisas não iam bem), Talitta Sanchotene, Melissa, Stainam, Ricardo, Luciana Matos, Isabella Almeida, Nelson Kotowski e Jonice Oliveira.

Aos meus amigos de COPPETEC, Rógea Rocha, Fernanda Costa, Vanessa Carla, Luiz Fernando, Amanda Varella, ViniciUs Von Held, ViniciOs Pereira, Carla Góis, Rafael Martino, Nathalia Marassi, Matheus Wildemberg (o homem das cargas!), Marcelo Caniato, Felipe Leite, e em especial à Patrícia Curvelo (Tia Pats!) e Robson Rocha (apesar de me sacanear direto!), por me safarem sempre e por serem ótimos amigos.

Aos amigos da Pagadoria de Pessoal da Marinha (PAPEM), pelo incentivo mostrado durante estes quatro anos em que estive lá, Dona Ângela, CT Marita, CC Raquel, CC Queiroz, CT Márcio Rocha, CC Glória, Danton, Passarini, SG Elaine, SG

Jaqueline e em especial à SG Vivian e Carlos Fernando por me aturarem sempre que aconteciam problemas!

Às minhas amigas Renata Machado, Ravelly Machado e Danielle Machado, pela amizade, e principalmente por me aturarem até nos dias em que eu estava mais chato durante esses anos.

À Camille Furtado, a menina dos mil apelidos e minha eterna "chefa". Sempre disposta a ouvir meus problemas, desabafos, a me sacanear ou a me ajudar sempre que precisei. "Doida", saiba que pode sempre contar comigo para o que der e vier! Muito obrigado!

Aos meus eternos amigos da EDK5, Daniel Leitão, Eduardo Fernandes, Rafael Monclar, Caio Neumann, Higo Fernandes e Bruno Caputo.

Aos meus amigos de longa data Cristina Maretti e Thiago Guadelupe, que mesmo sem colaborar diretamente com este trabalho sempre se mostraram motivadores em qualquer assunto que eu estivesse envolvido.

Ao meu "padrinho" Alfredo e minha "madrinha" Regina que sempre me incentivaram a continuar estudando, mesmo quando eu me mostrava um pouco cansado de tudo.

À minha irmã Amanda, que apesar de ter nascido em outra família, é uma irmã de verdade, afinal, Deus te fez nascer longe só para me dar trabalho de te achar. Difícil falar de uma pessoa que se parece muito comigo, é como se eu estivesse falando isso para um espelho. Sempre disposta a me ajudar, desde a graduação até agora, e para a qual eu estou sempre disposto a ajudar também. Bem, só tenho a agradecer por você ser esta grande amiga e dizer que pode contar comigo para o que der e vier.

E, finalmente, agradeço a minha família. Aos meus pais por terem me dado tudo que uma pessoa pode desejar em termos de família: carinho, amor, estudo e acima de tudo dignidade para que eu pudesse ser a pessoa que sou hoje. Aos meus irmãos, Bruno e Lucas, que apesar de me encherem um bocado, estão sempre dispostos a ajudar (mesmo que reclamando um pouco!). À minha "Dindinha" Vânia, minha tia Deise e minha prima Larissa estarem sempre ao meu lado. E por último, mas não menos importantes aos meus avós Neli e Edésio, sempre me incentivando a estudar, trabalhar,

mesmo que com isso eu ficasse com menos tempo para passar ao lado deles. Se não podiam ajudar com conhecimento para escrever ou implementar nada, me ajudavam com o que mais importava: atenção e amor (mesmo eu não podendo dar a atenção que eles mereciam). Vó e Vô, muito obrigado! Aos meus avós Maria e Martiniano e ao meu padrinho Edgard, que se ainda estivessem entre nós, tenho certeza que estariam vibrando com mais esta conquista.

A Deus, por tudo que me deu durante todo esse caminho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# MININGFLOW: ADICIONANDO SEMÂNTICA A WORKFLOWS DE MINERAÇÃO DE TEXTO

Daniel Cardoso Moraes de Oliveira

Março / 2008

Orientadoras: Marta Lima de Queirós Mattoso
Fernanda Araújo Baião

Programa: Engenharia de Sistemas e Computação

O processo de descoberta de conhecimento em textos (ou simplesmente KDT) é uma área de pesquisa altamente ativa e de onde emergem uma série de algoritmos e implementações que são desenvolvidos com o intuito de oferecer um apoio mais completo a este processo de descoberta do conhecimento. O projetista de um processo de KDT muitas vezes se depara com uma enorme quantidade de recursos que devem ser gerenciados (algoritmos, programas, parâmetros, resultados obtidos anteriormente, por exemplo). O ciclo de KDT é então modelado através da composição de diversas tarefas em um *workflow*, que será usado como instrumento base para a condução dos experimentos. Nesta dissertação, apresentamos o *MiningFlow*, um ambiente de apoio ao processo de KDT que combina a utilização de tecnologias de *workflows* com uma ontologia de domínio visando oferecer todo o apoio necessário para a realização de experimentos de mineração de textos. Este suporte se dará no momento da modelagem de *workflows*, em sua execução e na análise de resultados obtidos, utilizando-se de recursos semânticos para obter este apoio. Nossa implementação acoplou uma ontologia para o processo de KDT aos Sistemas Gerenciadores de *Workflows* Científicos Kepler e Taverna para executar os *workflows* gerados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MININGFLOW: ADDING SEMANTICS TO TEXT MINING WORKFLOWS

Daniel Cardoso Moraes de Oliveira

March / 2008

Advisors: Marta Lima de Queirós Mattoso
Fernanda Araújo Baião

Department: Computer and Systems Engineering

Text Mining (or Knowledge Discovery in Texts - KDT) is a very active research area and innumerous algorithms have been designed to support its discovery cycle. The designer is faced with several resources (e.g. algorithms, programs, parameters, datasets, results visualization) to manage. The KDT cycle is thus designed by composing several tasks in a *workflow*, which is used as a basis for conducting experiments. In this paper we present MiningFlow, a KDT environment that combines *workflow* technology with a domain ontology to support *workflow* modeling and result visualization to offer semantic support for KDT. Our implementation couples a KDT ontology to Kepler and Taverna *workflow* platform to execute KDT *workflows*.

SUMÁRIO

LISTA DE FIGURAS

# LISTA DE TABELAS

# Capítulo 1 Introdução

## 1.1 Motivação

Desde a última década do século passado, o processo e os mecanismos que permitem armazenar informações digitais têm se tornado extremamente baratos. Discos rígidos, memórias, entre outros dispositivos se tornaram muito mais populares do que eram no início dos anos 80. Juntamente com o advento da *Internet*, estes dispositivos colaboraram com a proliferação de documentos disponíveis para leitura e acesso. Entretanto, a taxa com que as pessoas absorvem a informação e a transformam em conhecimento se manteve a mesma, apesar da quantidade de informação disponibilizada ter crescido em escala exponencial. Além disto, grandes quantidades de informação contidas nestes documentos não são trivialmente visualizadas ou identificadas, necessitando de técnicas avançadas para que estas possam ser captadas, processadas, interpretadas e traduzidas em conhecimento que possa ser compreendido por seres humanos.

Mineração de textos (do termo em inglês *Text Mining*) é uma das áreas de pesquisa que endereçam este problema de "excesso" de informação disponível × capacidade limitada de absorção da informação. De fato, o texto bruto em si não quer dizer, necessariamente conhecimento que possa ser aproveitado. A quantidade de informação que temos disponível para análise (independente do domínio de conhecimento em que se esteja trabalhando) é tão volumosa que não teremos capacidade de absorvê-la nunca, uma vez que enquanto estamos absorvendo uma gama de informação, outra grande quantidade já está sendo gerada.

A área de mineração de textos foca justamente neste problema que é percorrer todo este condensado de informação para que se possa separar o que é útil e o que não é, o que pode ser transformado em conhecimento e o que não se pode. De um ponto de vista simplista, a solução é implementar um sistema computacional capaz de organizar, filtrar e retirar o que é valioso, mas na prática isto não é algo trivial de ser alcançado. Estudos nas áreas de inteligência artificial, redes neurais, sistemas inteligentes entre outros, vêm continuamente sendo realizados para implementar esta solução.

De acordo com o *Text Mining Research Group* da Universidade de Waikato na Nova Zelândia (TMRG, 2008), "Mineração de textos é a **procura por padrões** em um texto em **linguagem natural** e pode ser definido como o processo de análise do texto para extrair informação dele para um propósito particular".

A área de mineração de textos utiliza técnicas já consagradas de mineração de dados juntamente com técnicas de busca e recuperação da informação para prover novas soluções para problemas como extração de conhecimento de grandes coleções de textos (uma biblioteca médica ou de bioinformática, por exemplo). Estas técnicas se referem à captação de textos em linguagem natural, transformação dos mesmos em estruturas manipuláveis para, enfim, apresentá-las em uma estrutura formal.

Um exemplo de uma técnica consagrada de mineração de dados é a chamada categorização, que ao ser aplicada no âmbito de mineração de textos se propõe a identificar uma classe ou categoria a que um determinado documento pertencente dentro da coleção. Os algoritmos de categorização (e suas implementações) são focados em descobrir uma função que mapeie um conjunto de documentos em rótulos categóricos predefinidos, denominados classes. Uma vez descoberta esta função, a mesma pode ser aplicada a novos documentos de forma a prever a classe (ou categoria) em que tais documentos irão se encaixar.

Um outro exemplo de uma técnica consagrada de mineração de dados que pode ser aplicada a textos é a sumarização. Algoritmos deste tipo têm como finalidade buscar e identificar características comuns em documentos de uma coleção. Um exemplo da aplicação deste tipo de técnica é a busca por características em prontuários médicos. Digamos que um hospital tenha o prontuário de milhares de pacientes em meio eletrônico. Através da aplicação desta técnica, conhecimento do tipo "76% dos homens entre 25 e 40 anos que fumam e bebem têm câncer no futuro" pode ser obtido. Este tipo de conhecimento não é simples de ser visualizado (devido ao tamanho da massa de dados) e pode ser útil em casos de campanhas preventivas, por exemplo.

De acordo com Han e Kamber (2001), podemos dividir o processo de descoberta de conhecimento em textos (comumente referido somente como processo de mineração de textos) em três fases principais: **pré-processamento, mineração de textos e pós-processamento.**

A fase de **pré-processamento** é responsável por aplicar técnicas de limpeza e tratamento dos dados para as etapas que se seguem. Em geral, esta etapa consiste em remover o que for desnecessário para o entendimento do texto e à aplicação dos algoritmos de mineração. A fase de **processamento** é onde ocorre a busca efetiva por conhecimentos úteis através da aplicação de técnicas de mineração, como por exemplo, a categorização e a sumarização de textos, descritas anteriormente. O resultado desta fase é a geração de modelos que irão representar o conhecimento obtido ou disponibilizar um meio para que este conhecimento seja obtido. Regras (como as descritas no exemplo de sumarização) ou funções de categorização são exemplos dos produtos desta fase. A aplicação do modelo pode variar de acordo com o caso. Por exemplo, no caso de categorização, o modelo gerado receberá como entrada um documento, e, através das palavras que compõem o documento, a função de categorização (que foi o fruto da aplicação dos algoritmos de categorização) consegue identificar a qual classe o documento está associado. A fase de **pós-processamento** é responsável por validar os modelos obtidos pela fase de mineração (através da aplicação destes modelos em uma massa de dados cuja saída do modelo seja conhecida, para que se possa verificar sua veracidade), No caso da categorização, por exemplo, podemos utilizar como entrada do modelo documentos cujas categorias são previamente conhecidas para que se possa saber se o modelo é correto ou não (ou pelo menos tem um índice de acerto considerado aceitável). Além disto, a fase de pós-processamento pode ser a responsável por gerar a visualização de um modelo, por exemplo.

Como podemos perceber, o processo de mineração de textos possui etapas bem conhecidas e definidas, e que possuem uma seqüencia lógica entre si (uma vez que não faz sentido em um processo de descoberta de conhecimento em textos aplicar algoritmos de pré-processamento após a fase de mineração de textos, por exemplo).

Para cada etapa do processo de mineração de textos existem diversas tarefas, métodos, algoritmos, e *softwares* alternativos que podem ser aplicados. Para uma categorização de textos, por exemplo, tarefas de classificação de textos ou de agrupamento de textos podem ser aplicadas. Dentro da classificação de textos, vários algoritmos estão disponíveis como, por exemplo, árvores de decisão. Na seqüência, várias implementações de um mesmo algoritmo de árvore de decisão também podem ser escolhidas. A eficácia da tarefa, algoritmo e *software*, a serem escolhidos, depende muito da característica dos textos dos documentos a serem analisados. Na maioria das

vezes, várias avaliações precisam ser conduzidas até fazer a escolha mais acertada para os documentos em questão.

Desta forma, os analistas de mineração de textos se deparam com um cenário, muitas vezes complexo, em que os mesmos necessitam definir seqüências de tarefas compatíveis e escolhas adequadas de tarefas, algoritmos e *softwares* para serem usados, implementados e avaliados em cada fase do processo de descoberta do conhecimento em textos (pré-processamento, mineração e pós-processamento).

Para os algoritmos disponíveis, diversas bibliotecas de ferramentas e programas têm sido desenvolvidos, inclusive em código aberto (WEKA, 2007). Uma vez que o *software* é escolhido, o analista ainda deve se preocupar com uma série de tópicos que englobam definição de parâmetros para os *softwares*, documentos ou coleções de documentos que serão usados como entrada, entre outras decisões importantes a serem tomadas.

Por exemplo, a validação cruzada (HAN e CAMBER, 2001) é uma técnica comum aplicada no campo de mineração de textos. Nesta técnica, propõe-se que fases do processo de mineração de textos sejam executadas diversas vezes, variando-se alguns parâmetros de entrada. Desta forma, o processo será executado muitas vezes, variando-se apenas o parâmetro de entrada com o objetivo de se analisar a variação dos resultados finais.

Mesmo quando é encontrada a solução ideal, nada garante que numa próxima mineração de textos, o processo será facilitado. Em geral, esse processo de escolha não fica registrado. Resultados de mineração obtidos são armazenados sem uma associação ao processo que os geraram. Aproveitar experiências se torna quase impossível. Como conseqüência muito re-trabalho ocorre até em uma mesma equipe de analistas. Ferramentas auxiliares facilitadoras de tarefas são freqüentemente desenvolvidas, mas podem ficar desconhecidas para outros analistas. Ou ainda, *softwares* podem estar disponíveis, mas sem uma descrição que facilite a descoberta de seu propósito pelos analistas de mineração de textos.

Denomina-se um experimento de mineração de textos este conjunto de escolhas, avaliações e descrições envolvidas nas fases do processo de mineração de textos que possui como resultado a extração do conhecimento. Um experimento de mineração de

textos pode necessitar variar as atividades da seqüência de tarefas, os parâmetros de entrada e comparar resultados obtidos. Este tipo de experimento que é executado totalmente em um ambiente computacional é chamado de experimento *in-silico*. A condução do experimento necessita de apoio tecnológico em todo o seu ciclo de vida.

Os experimentos de mineração de texto vêm sendo realizados de modo *ad-hoc*, ou seja, não há sistematização do processo de mineração de texto, nem um sistema de apoio às decisões e avaliações ao longo do seu ciclo de vida. Estas decisões podem se tornar ainda mais complexas, dependendo da implementação dos algoritmos. Muitos *softwares* executam mais de uma atividade do ciclo de vida de um experimento de mineração de texto e as atividades englobadas nem sempre são documentadas. Desta forma, pode se tornar complicado saber qual serviço ou programa cobre qual atividade do processo.

Após a definição de tarefas, algoritmos e *softwares* do experimento pelo analista, é chegada a hora de instanciá-lo, fase na qual os programas escolhidos são efetivamente executados. Usualmente, estes programas são executados isolados uns dos outros, o que pode se tornar um fator complicador. Uma vez executado o experimento definido, o analista deve verificar se os resultados obtidos são satisfatórios, a fim de refinar o experimento, por exemplo. Porém, após uma seqüência $x$ de execuções, pode ser complicado para o analista relacionar todas as execuções de um "mesmo" experimento.

O desenvolvimento de um experimento, que reflita o ciclo de vida de mineração de textos, pode envolver diferentes plataformas associadas aos vários passos do processo, como o pré-processamento de coleções de documentos, remoção de *stop words* em um *cluster*, *stemming* e visualização dos resultados utilizando um sistema complexo de visualização. Conforme a quantidade de pesquisadores, desenvolvedores, implementações e propostas aumentam, trabalhar desta forma (através de *scripts* e sem documentação do processo) pode se tornar um processo caótico. Bibliotecas de *softwares* bem conhecidas da área como o RapidMiner (MIERSWA et al, 2006) ou o Weka (WITTEN e FRANK, 2005) possuem inúmeras ferramentas, mas são apenas repositórios de programas e algoritmos sem apoio para o ciclo de vida do experimento.

Podemos definir um processo caótico como aquele onde não existe controle por parte do analista (ou de um responsável pelos experimentos) em relação aos diversos

experimentos que são (ou já foram) executados. Sem este tipo de controle, o analista de mineração de textos desperdiça uma série de oportunidades como aproveitar resultados obtidos com experimentos anteriores ou identificar experimentos que sejam análogos ao que está sendo definido. O reaproveitamento de resultados anteriores se torna uma vantagem no processo de mineração de textos, uma vez que boa parte das implementações e de algoritmos existentes pode ser lenta já que manipulam uma quantidade muito grande de dados.

As alternativas a serem manipuladas pelo analista normalmente não são documentadas ou estruturadas e além disto, não há um registro dos experimentos que são executados e nem dos resultados intermediários que são obtidos nas execuções destes experimentos. Estes resultados intermediários aliados a informações referentes à execução e origem dos dados são denominados proveniência. Proveniência (BUNEMAN, KHANNA e TAN, 2001) é o processo de se armazenar a origem dos dados de cada experimento científico. Saber quem executou um determinado experimento, quais foram os resultados intermediários, os parâmetros de entrada, etc. são requisitos fundamentais para qualquer experimento científico. Um experimento só pode ser considerado válido caso a origem dos seus dados possa ser identificada e caso o mesmo possa ser refeito. Sem proveniência, estas duas características não podem ser atingidas.

Outro problema que podemos encontrar em um ambiente como este é a falta de semântica no processo. A definição de um experimento de mineração de textos pode ser uma tarefa razoavelmente simples para um especialista no assunto. Entretanto, quando nos referimos a usuários não-especialistas, a definição de um experimento pode ser uma tarefa nada trivial. O processo necessita, então, possuir mais semântica, de forma que possa auxiliar usuários não-especialistas a construírem um experimento e melhor documentar processos que já foram previamente definidos.

Este cenário pode ser observado em diversos tipos de ambientes, como por exemplo, em empresas que apliquem mineração de texto em suas atividades do cotidiano ou em laboratórios de pesquisa em mineração de texto. Estes laboratórios, em especial, possuem equipes com um número considerável de participantes que, em geral, interagem entre si e necessitam colaborar no desenvolvimento de experimentos de mineração de textos. Faz-se necessário, então, que exista controle de forma que os

experimentos realizados por um usuário possam ser devidamente avaliados ou reutilizados por outros para que não ocorra re-trabalho desnecessário.

O mesmo problema ocorre comumente em ambientes universitários. Por exemplo, muitos professores observam que muitas das teses, dissertações ou trabalhos de conclusão de curso possuem pontos de implementação em comum. Por exemplo, uma tese de um aluno focava em desenvolver um algoritmo de classificação $\alpha$, enquanto outro aluno estava desenvolvendo um algoritmo de classificação $\beta$. Apesar dos dois algoritmos serem diferentes, ambos utilizavam os mesmos algoritmos de pré-processamento para preparar a coleção de documentos para mineração. Os esforços de implementação estavam sendo desperdiçados, uma vez que cada aluno era obrigado a implementar sua fase de pré-processamento própria.

Com base nesses problemas envolvidos na concepção, execução, análise e documentação de um experimento em mineração de texto, podemos caracterizar o contexto global do objetivo desta dissertação de mestrado como sendo o de apresentar uma solução para apoiar o analista nas diversas etapas do ciclo de vida de um experimento em mineração de textos.

Problemas semelhantes aos aqui descritos para o apoio ao experimento de mineração de texto vêm sendo abordados pela comunidade de *workflows* científicos para diversas áreas como bioinformática, astronomia, e física de altas energias. De acordo com Mattoso (2006), a composição de processos num fluxo que encadeia as diversas análises de dados pode ser chamada de *workflow*, e ainda, que podemos chamar de "experimento científico" todo o processo de modelagem, execução e análise de resultados de *workflows* científicos. Inúmeros Sistemas de Gerência de *Workflows* Científicos (SGWf) vêm sendo propostos para apoiar o ciclo de vida de um experimento científico, dentre eles, Taverna (GOBLE e WROE, 2004), Kepler (ALTINTAS et al., 2004), VisTrails (FREIRE et al, 2006) ou o Triana (2007).

Embora o processo de mineração de texto seja muito distinto de bioinformática e astronomia, por possuir um ciclo bem definido (com etapas bem definidas e conhecidas), o mesmo pode ser visualizado como um *workflow* científico. Conseqüentemente, seus passos (atividades) poderiam se beneficiar dos SGWf e de toda

pesquisa e desenvolvimento para experimentos científicos que já foi concebida até o momento.

Realizamos diversos estudos experimentais com as ferramentas Kepler, Taverna e VisTrails, onde puderam ser verificadas vantagens, como editores gráficos para composição das tarefas do *workflow* (encontrados em todos os SGWf analisados), mecanismos de captura a análise de dados de proveniência (no caso do Taverna), mecanismos complexos de visualização e comparação de resultados (no caso do VisTrails), entre outros,.

Entretanto, para apoiar a concepção, análise e documentação de um experimento associado ao domínio já bem definido da mineração de textos, o uso desses SGWf não se mostrou efetivo. Consideramos fundamental o apoio não apenas ao *workflow* em execução, mas principalmente ao ciclo de vida do experimento. Os sistemas existentes não associam diversas execuções de *workflows* como fazendo parte de um único experimento. O VisTrails permite a comparação entre os resultados de um mesmo *workflow* que foi executado com variação de parâmetros. Porém, o VisTrails não possui o conceito de experimento científico, que permitiria, por exemplo, a associação de outras definições de *workflow* a um mesmo experimento. Consideramos ainda o apoio de uma ontologia de domínio essencial para o experimento desta natureza. Além de não termos encontrado uma ontologia definida para mineração de textos em nenhuma destas ferramentas, não encontramos, em nenhum dos SGWf analisados, o apoio à criação de uma ontologia e sua associação ao experimento. O sistema que mais se aproxima deste apoio semântico é o Taverna/myGrid. Entretanto, na forma em que está disponível, o sistema se restringe a sua própria ontologia (no domínio da bioinformática).

Outro requisito do apoio ao experimento de mineração de textos é que, devido ao tempo de execução de algumas tarefas ser demasiadamente grande (dependendo do tamanho da coleção com a qual estamos trabalhando), o sistema deve armazenar os dados intermediários da execução, para que estes possam ser posteriormente recuperados. Dos SGWf analisados, apenas o Taverna possui esta funcionalidade de forma nativa e operacional. Como será discutido no capítulo 2 desta dissertação, os outros SGWf analisados possuem apenas propostas para implementar esta funcionalidade, mas nada liberado para uso ou 100% operacional.

Consideramos também que o apoio semântico ao experimento de mineração não deva exigir o uso de um SGWf específico. Os analistas devem ter liberdade de escolha do SGWf de acordo com seus requisitos operacionais sem perder a semântica do experimento.

## 1.2 Objetivos

O objetivo desta dissertação é apresentar uma solução para oferecer apoio ao analista de mineração de textos durante o ciclo de vida de seu experimento (definição, execução e análise e documentação de resultados). Visamos uma solução que atenda as especificidades do domínio da mineração de textos e que aproveite as vantagens dos SGWf existentes, ao mesmo tempo em que não cria dependência desses sistemas.

Para tanto, propomos o *MiningFlow*, um sistema que provê o uso combinado de experimentos científicos (com a utilização de tecnologias de *workflow*) com uma ontologia de domínio para mineração de textos. O *MiningFlow* atua como uma camada intermediária (*front-end*) entre o usuário analista de mineração de textos e o SGWf, de forma a prover ferramentas com semântica para o registro e a descoberta de recursos (leia-se programas, serviços, *scripts*) durante o processo de mineração de textos, especificamente na composição das atividades do *workflow* (durante a modelagem de experimentos) e na sua execução.

Apresentamos também neste trabalho uma ontologia voltada para o domínio de mineração de textos (*MF-Ontology*) baseada e estendida da proposta de Cannataro e Comito (2003). Esta ontologia mapeia e descreve os principais conceitos e relacionamentos entre os aspectos-chave do domínio de mineração de textos.

O *MiningFlow* foi implementado baseado em serviços *web* e foi construído para ser independente de qualquer SGWf. Em nossos experimentos pudemos avaliar essa independência ao utilizamos os SGWf Taverna e Kepler nos estudos de observação e no estudo de caso utilizado para a validação da proposta.

Assim sendo, absorvemos características importantes de SGWf, como a utilização de ontologias do Taverna, para oferecer semântica ao processo de definição do *workflow*, a capacidade do VisTrails de associar diversas execuções de um mesmo *workflow* com a finalidade de comparar resultados, o mecanismo de proveniência do

Taverna para armazenar os resultados intermediários, além de incorporar novas funcionalidades como a definição em mais alto nível do *workflow*.

No *MiningFlow*, o analista de mineração de texto possui recursos semânticos que o ajudam a definir seu experimento de mineração de textos, tanto em alto-nível (baseada nos conceitos que estão definidos e mapeados na ontologia proposta) quanto em mais baixo nível (em termos de programas ou serviços executáveis) com o apoio da *MF-Ontology* durante o processo de definição dos *workflows*. Além disso, os dados de proveniência são associados aos conceitos da ontologia, fazendo com que as informações de origem e de execução dos *workflows* contenham mais semântica.

Além disso, a proposta provê meios para que o analista recupere informações de execuções de experimentos anteriores e possa realizar análises posteriores que forneçam subsídios necessários para a definição de novos experimentos.

Os resultados dos testes realizados com o *MiningFlow* apresentaram evidências quanto à efetividade dos recursos disponíveis aos analistas no apoio às atividades de definição, execução e análise dos resultados dos experimentos em mineração de textos via ferramenta. Entretanto, algumas observações pertinentes foram realizadas durante o estudo de caso e são detalhadamente descritas no capítulo referente à validação da proposta.

O estudo de caso da ferramenta foi realizado com alunos de mestrado e doutorado do Programa de Engenharia de Sistemas (PESC) da COPPE-UFRJ e do Programa de Engenharia Civil (PEC) da COPPE-UFRJ. Os serviços disponibilizados na ferramenta para validação foram adaptados de implementações disponibilizadas por alunos do PEC especializados no domínio de mineração de textos.

*MiningFlow* encontra-se disponível para acesso em www.cos.ufrj.br/~danielc. No próprio portal, o usuário estará apto a criar uma nova conta que será autorizada para acesso.

## 1.3 Organização dos Capítulos

Para facilitar o entendimento desta dissertação, o capítulo dois oferece uma revisão do processo de mineração de texto bem como dos conceitos relativos a *workflows*. Além disso, são mostrados trabalhos correlatos nesta área.

O capítulo três destina-se a explicar a ontologia de mineração de textos que foi desenvolvida, desde o processo de modelagem até o momento da validação. A arquitetura proposta para o *MiningFlow* e a explicação de cada um dos seus módulos se encontra no capítulo quatro. O capítulo cinco descreve a implementação do sistema proposto, enquanto que o capítulo seis apresenta a metodologia utilizada para o estudo de observação ao qual o sistema foi submetido e as análises geradas a partir deste estudo. O capítulo sete se propõe a apresentar a conclusão deste trabalho, bem como os possíveis melhoramentos do mesmo e apontar futuros trabalhos correlatos.

# Capítulo 2 Apoio ao Ciclo de vida de Experimentos de Mineração de Textos com Gerência de *Workflows*

Segundo GOI (2007), estima-se que 80% das informações disponíveis na *web* são armazenadas em documentos textuais. Estes documentos, eventualmente, contém informações intrínsecas que podem ser aproveitadas pelos diversos ramos da indústria ou de pesquisa. A captação destas informações, sua transformação e a geração de conhecimento útil a partir dela são objetivos endereçados pelo que chamamos de processo de descoberta de conhecimento em textos (ou simplesmente processo de mineração de textos).

Como apresentado na introdução desta dissertação, uma das contribuições deste trabalho é a combinação de tecnologias de *workflows* no apoio ao ciclo de vida de um processo de mineração de texto com suporte semântico através de ontologias.

Neste capítulo descrevemos alguns conceitos importantes que serão referenciados ao longo da dissertação. Alguns conceitos como o ciclo de vida de um experimento de mineração de textos, ciclo de vida de um *workflow* científico, serviços *web* (*web services*), conceitos e padrões de *workflows* científicos e sistemas gerenciadores de *workflows* científicos (SGWf), considerados conceitos-chave para o entendimento desta dissertação.

Na última seção deste, dissertamos a respeito de trabalhos relacionados que possuem (em maior ou menor quantidade) características semelhantes com a proposta *MiningFlow* e a partir dos quais nos baseamos para construí-la.

## 2.1 Processo de Mineração de Texto

### 2.1.1 Definição e Objetivos

Mineração de textos pode ser definida como uma área de pesquisa que lida com grandes quantidades de textos escritos em linguagem natural e não-estruturada, com o objetivo final de combinar técnicas já consagradas de mineração de dados e busca e

recuperação da informação (entre outras) para extrair informação útil desta grande massa de dados.

Segundo Feldman e Sanger (2006) podemos fazer uma analogia com o processo de mineração de dados, definindo mineração de textos como: "um processo que visa extrair informação útil de um certo volume de dados (documentos) através da identificação e exploração de certos padrões não triviais". Devido sua natureza não estruturada, o processo de mineração de textos é uma tarefa que exige um tratamento diferenciado do processo de mineração de dados em geral.

Segundo GOI (2007), estima-se que 80% das informações disponíveis na *internet* são armazenadas em documentos textuais e que cerca de 80% destas informações se encontram em língua inglesa. É quase certo que nesta imensa massa de dados exista informação não-trivial que possa ser extraída. Os e-mails de clientes, anotações de *call center*, respostas abertas de pesquisas, formulários da *web* e outras fontes de texto que uma empresa, instituição ou laboratório coletam, contém até quatro vezes mais dados importantes do que seus dados estruturados armazenados. Isso significa que projetos de mineração de dados focados apenas em dados estruturados utilizam (estatisticamente considerando) apenas vinte por cento das informações disponíveis enquanto que projetos focados em mineração de textos atingem uma massa muito maior de dados.

Entretanto, o desenvolvimento de algoritmos de processamento de texto envolve uma série de dificuldades, uma vez que, em geral, para cada experimento de mineração de texto que se está trabalhando há uma etapa de pré-processamento específica. A etapa de pré-processamento (principalmente as atividades que a compõem) pode variar de acordo com a coleção de documentos com a qual está se trabalhando e de acordo com o objetivo final e com os algoritmos que serão utilizados na fase de mineração. As etapas do processo de mineração de textos serão mais bem descritas na seção 2.1.2 deste capítulo

Além disso, a particularidade de cada idioma deve ser contemplada no desenvolvimento da aplicação. A implementação de um processo de mineração de texto para a língua portuguesa difere em diversos pontos de um processo que esteja aplicado em textos de língua inglesa ou chinesa, por exemplo. Um exemplo desta diferença são

algoritmos de pré-processamento que removem *stop words* (palavras que são comuns em textos e que não são consideradas pelos algoritmos de mineração). Estas palavras variam de idioma para idioma. Além disso, a atividade de *stemmização* também varia de idioma para idioma, visto que a redução de radicais não é igual em português e inglês, por exemplo.

### 2.1.2 Ciclo de Vida do Processo de Descoberta de Conhecimento em Textos

Para que possamos entender melhor o processo de descoberta de conhecimento em textos, devemos compreender como está definido o ciclo de vida de um experimento deste gênero. Em geral, a literatura divide o processo de mineração de textos em três grandes etapas: pré-processamento, mineração de textos (ou o processamento propriamente dito) e o pós-processamento.

A etapa de **pré-processamento** é onde são executadas funções que visam analisar e padronizar o texto de forma que a etapa seguinte (mineração de textos) já não tenha necessidade de trabalhar com um texto "bruto". Como em experimentos desta natureza visa-se trabalhar com textos escritos em linguagem natural (ou seja, sem padrão algum), algumas técnicas devem ser aplicadas para diminuir a quantidade de informação "inútil" para as etapas que se seguem. Por exemplo, retirar as *stop words* do documento, de forma que palavras sem muito "peso" no significado do documento, não sejam nem analisadas em etapas posteriores, ou ainda reduzir as palavras do documento aos seus radicais (*stemmização)*, por exemplo. Algumas descrições das técnicas de *stemming* podem ser encontradas em Moreira e Huyck (2001), Porter (1980) e Lovins (1968). Podemos considerar o tratamento da informação no processo de mineração de textos uma das partes mais importantes, pois seu desempenho está diretamente relacionado ao resultado da aplicação das técnicas de mineração da fase de processamento (mineração de textos), uma vez que, se permitirmos que dados não necessários sejam minerados, o processo pode se tornar lento ou até improdutivo, já que normalmente as coleções de documentos possuem um volume muito grande.

A etapa de **mineração de textos** é a etapa-foco do processo, uma vez que na mesma são aplicadas técnicas e algoritmos de mineração para gerar os modelos referentes aos dados de entrada, ou seja, o produto final que se deseja. Nesta etapa do

processo é que são desenvolvidas aplicações e componentes que visam implementar diversos algoritmos para o processamento de coleções de documentos nas mais diversas tarefas de mineração de textos. Como produto desta etapa, são identificados padrões, modelos, regras ou funções que nos permitem extrair ou visualizar o conhecimento que normalmente se encontrava intrínseco nos textos dos documentos. Alguns exemplos de tarefas consagradas que são utilizadas nesta fase do processo são:

**Classificação** ou **Categorização**: técnica que possui foco na descoberta de uma função que possa mapear um conjunto de documentos para um conjunto de rótulos, ou categorias. Inicialmente os algoritmos deste tipo são aplicados a dados cujas categorias são conhecidas, para que se possa avaliar a qualidade da função que foi gerada. Uma vez obtida a função, a mesma pode ser aplicada a novos documentos de forma a prever a classe que tais documentos se enquadra.

**Clusterização** ou **Agrupamento**: atividade cujo objetivo é separar os documentos de uma coleção em subconjuntos ou *clusters*, de tal forma que os elementos de um mesmo agrupamento compartilhem características comuns. Assim como na clusterização em mineração de dados, é um objetivo maximizar a similaridade *intracluster* e minimizar a similaridade *intercluster*.

**Sumarização**: esta tarefa, comum em mineração de dados, tem como objetivo de identificar características comuns entre os dados (no caso, entre os termos do documento). Por exemplo, em uma grande coleção de contratos de uma editora, após aplicarmos algoritmos de sumarização, características como "Assinam a revista *x*, homens entre 25 e 30 anos com nível superior e sem filhos", entre outras.

Definições e explicações detalhadas sobre as técnicas anteriormente descritas podem ser encontrados em Appelt e Israel (1999), Yang, e Pedersen (1997), Chidanand e Damerau (1994), Vasileios, Gravano e Maganti (2000), Kosala e Blockeel (2000), Borko e Bernick (1963) e Berry (2004).

A etapa de **pós-processamento** é a responsável por preparar os modelos e padrões obtidos na etapa de mineração para o entendimento e validação do usuário. Funções de visualização, comparação de resultados e validação de modelos estão englobadas nesta etapa do processo. A figura 2.1 ilustra as etapas do processo de mineração de textos.

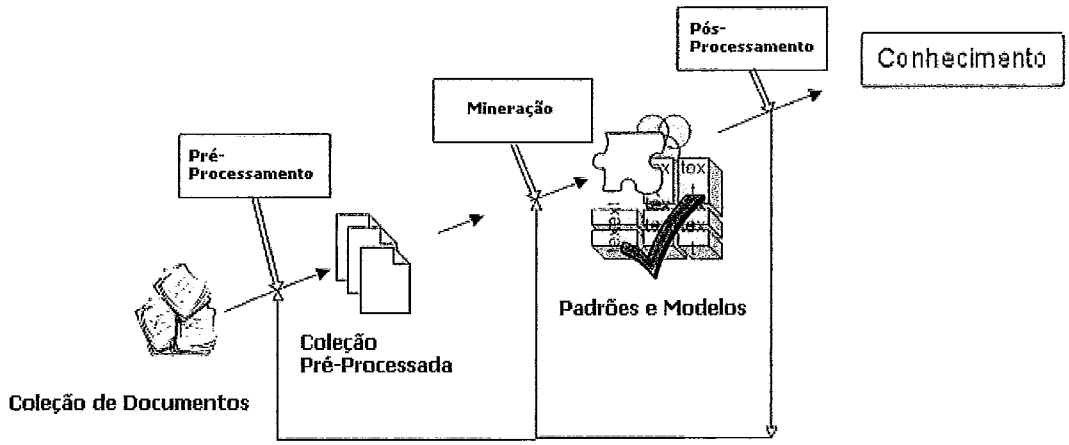


**Figura 2.1 Esquema gráfico do processo de mineração de textos.**

Por se tratar de um processo iterativo e interativo, as fases anteriormente explicitadas podem ser repetidas quantas vezes seja necessário. Por exemplo, ao validar um modelo obtido como resultado, o analista de mineração de texto pode não ter atingido o objetivo necessário e pode necessitar re-executar a atividade de mineração de dados alterando alguns parâmetros para avaliar e comparar os resultados obtidos (e suas possíveis variações).

### 2.1.3 Aplicações Computacionais de Suporte ao Processo de Descoberta de Conhecimento em Textos

Existem inúmeras aplicações disponíveis (e muitas delas de código aberto) que oferecem suporte ao processo de mineração de dados / textos. Entretanto, muitas destas aplicações não visam atender especificamente o processo de mineração de textos e sim o de mineração de dados. Diferentemente do Processo de descoberta de conhecimento em textos, o processo de KDD (Descoberta de Conhecimento em Bancos de Dados) foca em lidar com dados estruturados (tuplas em uma base de dados, por exemplo) e padronizados. Desta forma, muitos dos algoritmos de KDD não podem ser diretamente aplicados no processo de descoberta de conhecimento em textos, pois o mesmo não manipula dados estruturados, uma vez que os documentos são escritos em linguagem natural.

Além disto, muitas das ferramentas trabalham com formatos proprietários ou não atendem a todas as especificações necessárias (podem não cobrir o processo de mineração de textos como um todo, por exemplo). Algumas das ferramentas estudadas (que serão detalhadas a seguir) nem chegam a atender o processo de descoberta de conhecimento em textos como um todo, focando somente na etapa de mineração de textos. As fases de pré-processamento e pós-processamento, por muitas vezes nem chegam a serem cobertas pela ferramenta.

Obviamente, o analista de mineração de textos não necessita obrigatoriamente utilizar uma destas ferramentas em seu trabalho. Muitos dos pesquisadores ainda preferem trabalhar com suas próprias implementações sem acoplá-las a nenhum arcabouço (*framework*) existente. Esta solução, apesar de se mostrar mais prática em uma primeira análise, pode acarretar em problemas quando a equipe de desenvolvimento das aplicações e seus usuários aumentam. Problemas como a falta de controle das implementações existentes, verificação de equivalência entre implementações, registro de experimentos que foram realizados (o que os faz complicados de serem reproduzidos ou compartilhados), as próprias definições dos experimentos não se encontram em um ambiente comum que possa ser acessado por todos, entre outros problemas.

Sem um arcabouço definido, todos os experimentos devem ser executados por meio de roteiros (*scripts*) pré-definidos, e trabalhando desta forma, o analista eventualmente consegue ter registro dos resultados intermediários e do próprio experimento, assim como está apto a realizar a análise comparativa dos resultados finais, registro dos parâmetros de entrada utilizados, entre outros requisitos. Entretanto, o problema é que este registro é realizado de forma isolada e não padronizada, sem uma sistemática aplicada, o que faz ser quase impossível o compartilhamento das informações.

Imagine, por exemplo, os registros de dados intermediários de um grupo com diversos analistas. Se cada analista utilizar o padrão de nomes de arquivos intermediários como *<nome do experimento><mês><ano>*, diversos dados de diversos analistas podem estar denominados da mesma maneira. Além de que, se o nome do arquivo for modificado, toda a (pouca) semântica que existia foi perdida, sem que se possa saber a qual experimento este dado se referenciava.

Estes requisitos, apesar de não impedirem a execução de um experimento e nem de considerarem o mesmo como um experimento cientifico, são de suma importância para facilitar o trabalho do analista no momento de validar o seu experimento ou reproduzi-lo. Nenhum experimento científico tem validade caso não possa ser reproduzido, não possua um registro dos seus parâmetros de entrada e nem possa ser comparado a outros resultados obtidos. Se estes registros de dados intermediários, parâmetros e execuções forem realizados de forma sistemática e padronizada, muito do trabalho (e dos riscos que o analista corre, já que pode perder informações trabalhando de forma isolada) são diminuídos.

A seguir dissertamos sobre algumas das ferramentas de apoio ao processo de mineração de dados e textos estudadas durante o desenvolvimento deste trabalho.

### 2.1.3.1 InfoMiner

O *InfoMiner* (INFOMINER,2007) é uma aplicação para mineração de dados desenvolvida por pesquisadores da universidade *Otto von Guericke* de Magdeburg (Alemanha).

O *InfoMiner* possui uma interface gráfica na qual o cientista possa definir seu fluxo de atividades de forma simples. O *InfoMiner* não disponibiliza recursos importantes como armazenamento de dados intermediários ou mecanismos para auxiliar o analista no momento da definição de seu experimento. Outro ponto negativo da ferramenta é que não há nenhum componente ou módulo específico para lidar com mineração de textos.

### 2.1.3.2 Weka

O WEKA (2007), (Witten e Frank, 2005) ou *Waikato Environment for Knowledge Analysis* é um pacote de *software* desenvolvido em Java na Universidade de Waikato, Nova Zelândia. O Weka é uma "biblioteca" de *softwares* desenvolvidos sob a licença GPL (*General Public Licence*) sendo, portanto, passível de ter seu código fonte modificado ou estendido conforme a necessidade do usuário.

O Weka tem como objetivo principal agregar, em um único ambiente, algoritmos e suas diferentes implementações provenientes de diferentes abordagens do processo de mineração de dados.

Entretanto, o Weka não oferece suporte ao processo de mineração de textos, uma vez que o mesmo é voltado para o processo de mineração de dados. Cada passo do processo deve ser executado independentemente na ferramenta, o que pode ser um ponto de dificuldade para novos usuários.

Para facilitar o processo de definição do experimento, o pacote Weka incorporou o módulo *KnowledgeFlow* (WEKA, 2007) ao seu conjunto de ferramentas. O *KnowledgeFlow* é uma interface gráfica que permite que definamos nosso experimento de mineração de dados através de um fluxo, que chamaremos de *workflow* mais a seguir no texto, apenas conectando os componentes que representam funções de pré-processamento, tarefas de mineração de dados ou funções de pós-processamento. Este módulo, entretanto, está voltado para o processo de mineração de dados, não contemplando as necessidades do processo de mineração de textos. Além disto, o *KnowledgeFlow* não possui todas as características necessárias para atender um ciclo de vida de um experimento de mineração de textos, como por exemplo, o registro dos dados intermediários do experimento, documentação do processo de definição do experimento e dos resultados finais, mecanismos para auxiliar o analista no momento da definição do experimento, entre outros.

#### 2.1.3.3 RapidMiner (antigo YALE)

*RapidMiner* (anteriormente conhecido como YALE) (Mierswa et al., 2006) é um ambiente de mineração de dados que introduz o conceito de operadores aninhados (*nested operators*) para representar o processo de mineração de dados. O *RapidMiner* visa atender usuários com aplicações de mineração de dados complexas.

O *RapidMiner* incorpora diversas ferramentas, como o WEKA LibSVM (LIBSVM, 2007), e tem um padrão bem definido de interface entre os módulos (operadores). Esta ferramenta, apesar de estar voltada para o processo de mineração de dados também disponibiliza módulos de mineração de textos e mineração de dados distribuída.

O *RapidMiner*, apesar de oferecer uma grande gama de implementações, possui algumas desvantagens importantes. O conceito de operados aninhados não é utilizado em nenhuma outra ferramenta, logo, qualquer implementação que se queira acoplar ao *RapidMiner*, deve seguir este padrão pré-estabelecido. O *RapidMiner* não consegue executar serviços *web* e nem invocar outros códigos que tenham sido desenvolvido por usuários. Além disto, o *RapidMiner* não está voltado para o processo de mineração de textos (apesar de oferecer alguns componentes para mineração de textos, o sistema como um todo é voltado para mineração de dados) e não possui nenhuma facilidade para que o usuário defina e execute o seu experimento desta categoria.

#### 2.1.3.4 Text Mining for Clementine (TMFC)

O *software* de mineração de dados Clementine (CLEMENTINE, 2007) é desenvolvido pela empresa SPSS, e disponibiliza aos seus usuários uma série de ferramentas e componentes para mineração de dados e textos.

O módulo TMFC possibilita que o usuário combine no Clementine, uma ferramenta para lidar com dados não-estruturados e com os dados estruturados tradicionais, para aumentar significativamente o entendimento sobre seus clientes, por exemplo.

O TMFC identifica e extrai preferências e opiniões, que ajudam a criar modelos preditivos mais profundos e resultados mais precisos.

A mineração de textos pode ser utilizada em quase todas as situações de negócio ou pesquisa que envolvem dados não-estruturados. Aqui estão alguns exemplos de aplicações do TMFC em uma organização típica:

- O TMFC possibilita aos usuários descobrir informações não-triviais contidas em dados textuais sem que seja necessária experiência lingüística ou treinamento especial.

- A interface visual é um ponto muito forte na ferramenta, o que torna a mineração de texto direta e eficiente para os usuários de negócio.

- Além disto o TMFC disponibiliza algumas funcionalidades interessantes como o suporte ao *Predictive Modeling Markup Language* (PMML,

2007), que está se tornando um padrão do mercado, além de trabalhar com formatos de documentos-texto padrão como texto pleno, PDF, HTML e XML.

#### 2.1.3.5 Text Analyst

O *TextAnalyst* é uma ferramenta proprietária que tem como finalidade principal executar análise semântica de textos em diferentes domínios de aplicação. A ferramenta é baseada em uma tecnologia de redes neurais proprietária. O *TextAnalyst* se encontra entre as mais utilizadas ferramentas para mineração de texto do mercado e disponibiliza as seguintes funcionalidades (lista não exaustiva):

- Estruturação hierárquica dos documentos em tópicos e sub tópicos.
- Criação automática de uma base de conhecimento.
- Resumo automático de textos.
- Busca semântica de informações.
- Criação automática de índices.
- Entre outras.

Apesar de disponibilizar uma grande variedade de componentes e estar totalmente voltado para o processo de mineração de textos, o *TextAnalyst* é uma aplicação proprietária, e conseqüentemente, não está passível de alterações ou de ser estendida conforme a necessidade do usuário. Além disso, o *TextAnalyst* só executa componentes que estejam definidos segundo o seu padrão, não estando apto a executar implementações em padrões abertos como serviços *web*.

## 2.2 Serviços *web* (*web Services*)

Nesta subseção definiremos serviços *web*, pois este conceito será referenciado mais adiante neste capítulo e é um conceito importante para o entendimento desta dissertação. Podemos definir um serviço *web* como um serviço que se encontra disponível na *Internet* ou na *Intranet*, utiliza-se de um sistema padrão de mensagens

XML (W3C, 2006) e não está acoplado a nenhum sistema operacional ou linguagem de programação específica (CERAMI, 2002).

Em geral, os serviços *web* devem possuir duas características principais:

- Serem autodescritíveis: quando um serviço *web* é disponibilizado, ele deve disponibilizar também uma interface de acesso ao mesmo para que os consumidores do serviço possam integrá-lo à sua solução proposta. Esta autodescrição deve ser possível via uma gramática XML, e deve conter as funcionalidades do serviço, a localização do mesmo, os protocolos utilizados para comunicação entre outros pontos.

- Pode ser facilmente descoberto: devem existir meios de se pesquisar um serviço *web* na rede e encontrá-lo. Para isso devem existir catálogos que forneçam um modo simples de busca de serviços *web*.

Um adendo deve ser feito quanto à utilização da palavra *web* para designar os serviços. Os serviços não necessariamente devem estar disponibilizados na *Internet*. É comum que em grandes empresas ou em entidades de pesquisa, alguns serviços estejam apenas disponibilizados na rede interna da corporação, sem que usuários extracorporação tenham acesso a estes serviços. Nestes casos, costuma-se chamar a implementação apenas de "serviço", uma vez que a mesma não se encontra disponível na *web*.

Os serviços *web* estão baseados em três elementos principais que por sua vez são derivados do XML: o WSDL (W3CNOTE, 2007), uma linguagem padrão para descrever os serviços *web*, UDDI (2007) como catálogo de serviços (também chamado de diretório de serviços, onde se encontram as descrições dos serviços, para que os consumidores possam encontrá-los. Registros de serviços, como o UDDI, possuem implementações sofisticadas da operação de publicação) e o SOAP (W3CNOTE, 2007b) como protocolo de comunicação entre os serviços.

Como a própria definição de serviço *web* já nos mostra, o mesmo pode ser implementado em qualquer linguagem, tanto o consumidor quanto o provedor do serviço. Um consumidor pode ser implementado em Java no sistema operacional

Windows enquanto o provedor do serviço pode estar implementado em C++ no sistema operacional Linux.

## 2.3 *Workflows*

Nesta subseção do capítulo trataremos das tecnologias de *workflows*, explicitando os conceitos, ferramentas e requisitos necessários, além de contextualizar o processo de mineração de textos como sendo um *workflow* científico.

### 2.3.1 Definição e Objetivos

Em toda a literatura podemos encontrar diversas definições diferentes para o termo *workflow*. Entretanto, por questões de padronização, resolvemos assumir a definição de *workflow* dada pelo *Workflow Management Coalition* em WFMC (2007) onde "*workflow* é a automação de um processo de negócio, inteiro ou por partes, durante o qual documentos, informações e atividades são passadas de um participante para outro para que estes desenvolvam suas ações, respeitando um conjunto de regras procedimentais". Podemos definir *workflow* de uma forma mais simples como qualquer tarefa executada em série ou em paralelo por dois ou mais membros de um grupo de trabalho (*workgroup*) visando um objetivo comum.

Apesar da definição da WfMC ser abrangente, uma vez que considera qualquer tipo de fluxo de trabalho (o que nos leva a concluir que um *workflow* pode ser executado totalmente de forma manual), normalmente tecnologias da informação são aplicadas no *workflow* para que se possa ter uma automatização das atividades que os compõem, facilitando, ou muitas vezes simplesmente tornando viável, a execução das tarefas.

Justamente a aplicação de tecnologias de informação na elaboração de *workflows* torna esta tecnologia interessante. Se pensarmos em termos de serviços *web*, cada atividade do *workflow* pode ser mapeada para ser representada por um serviço *web*. Desta forma, o *workflow* (ou o *pipeline* de atividades) pode ser encarado como uma orquestração (composição) de diferentes serviços *web* com a meta de se atingir um resultado final. A utilização destas duas tecnologias de maneira integrada vêm sendo adotada em diversas pesquisas (KEPLER, 2007) (Freire et al., 2006) (Altintas et al.,

2004) (Goble, Wroe e Stevens, 2003) (TRIANA, 2007) e se firma como um ramo forte de pesquisa (Virdell, 2003).

Um comentário muito pertinente a ser feito nesta hora é que em grande parte dos casos e da literatura de negócios, os *workflows* são definidos como sendo um ferramental exclusivo para processos de negócio, mas não obrigatoriamente o são. Para comprovar tal comentário, apresentamos neste momento uma nova categoria de *workflows*: os chamados *workflows* científicos.

Nos *workflows* científicos, cientistas (dos mais diferentes domínios de conhecimento existentes) utilizam a tecnologia de wokflows para elaborar seus experimentos e desenvolver seus projetos de pesquisa. Segundo MEYER (2004) um grande diferencial entre esses dois tipos de *workflows* é que os cientistas normalmente especificam seus próprios *workflows* científicos enquanto que nos wokflows referentes a processos de negócio essa especificação é feita normalmente pelos administradores do sistema. Além disto, no caso dos experimentos de mineração de textos, além do analista/projetista do *workflow* necessitar elaborar seu próprio *pipeline*, ele poderá necessitar executar diversas vezes o mesmo *workflow*, mudando apenas alguns parâmetros de entrada e necessitará comparar os resultados obtidos através desta variação.

### 2.3.2 *Workflows* científicos

Em geral, os *workflows* científicos muito se assemelham a *workflows* para processos de negócio no que tange a sua forma e aos padrões de *workflows* ao quais ambos os tipos de *workflow* aderem (AALST e HEE, 2003).

Entretanto, os *workflows* científicos possuem uma série de requisitos que não são atendidos por *workflows* para processos de negócios de acordo com Singh e Vouk (1996), como por exemplo, a execução parcial de um *workflow*, armazenamento de dados de proveniência, entre outros.

Como foi dito anteriormente, no âmbito científico, muitas vezes é necessário que o próprio cientista desenvolva o seu *workflow*, enquanto em uma empresa existe um administrador que tem como tarefa definir o *workflow* que será executado por diversas pessoas dentro da empresa.

Além disto, em um ambiente científico é muito importante a re-execução dos *workflows* projetados, para que se possam validar os resultados obtidos, comparando-os com resultados gerados em execuções prévias do mesmo *workflow*. O armazenamento dos resultados gerados e de metadados que descrevam estes dados se faz muito importante.

A fase na qual o desenvolvimento de *workflows* científicos necessita de um suporte computacional maior provavelmente é a fase de definição do *workflow*. Esta definição normalmente se dá através de uma ferramenta gráfica que possibilite o desginer/projetista/analista a escolher as melhores tarefas para o seu *workflow* (lembrando que tarefas podem ser programas, serviços *web*, serviços em grade, entre outros).

Além disto, é de suma importância que exista alguma ferramenta ou módulo que apóie o analista de mineração de texto durante a elaboração do seu *workflow*, seja um apoio no momento de escolher os serviços a serem executados ou no momento de redefinir parâmetros de entrada, por exemplo.

É interessante que a definição do *workflow* cientifico possa se dar em termos de *workflows* abstratos e *workflows* concretos conforme definido por Cavalcanti (2003). Um *workflow* concreto é aquele onde as atividades que compõem a seqüência de passos do *workflow* se referenciam diretamente às instâncias dos serviços ou dos programas que serão executados. A definição concreta do *workflow* é a definição que será executada diretamente pela máquina de *workflows* do sistema gerenciador de *workflows* científicos (ou SGWf, como será explicado melhor nas próximas subseções).

Uma definição abstrata de um *workflow* é uma definição de mais alto nível do que a concreta. Nesta definição do *workflow*, as atividades que compõem os passos do *workflow* são representadas em termos de nomes lógicos ou conceitos. Esta representação pode ser utilizada para representar um *workflow* de mais alto nível ao qual o *workflow* concreto estará baseado no momento de sua definição. O *workflow* abstrato então é composto por uma série de passos que serão definidos por conceitos de mais alto nível que devem estar alinhados com os serviços que forem posteriormente escolhidos para compor o *workflow* concreto.

Como já podemos observar, uma série de requisitos novos se fazem necessários quando tratamos de *workflows* científicos. Uma relação completa das necessidades diferenciadas dos *workflows* científicos pode ser encontrada no trabalho de Weske, Vossen e Medeiros (1996). Os requisitos principais deste trabalho são reproduzidos a seguir:

- **Execução Parcial**: como em diversos experimentos científicos o tempo de execução algumas vezes é grande, execuções parciais (ou pausas na execução) são necessárias. O cientista também pode ter necessidade de verificar o andamento do experimento algumas vezes durante a execução.

- **Reutilização**: *workflows* (abstratos e concretos) anteriormente definidos devem poder ser reutilizados, de forma que o conhecimento que foi gerado não se perca.

- **Dados de Proveniência**: em um ambiente científico é importante conhecer de onde os dados gerados foram obtidos. Saber quem executou o experimento, quando e por qual razão são informações cruciais no âmbito cientifico para que o conhecimento gerado não seja invalidado. Um experimento científico só pode ser considerado válido quando pode ser re-executado. Desta forma, devemos manter um registro das execuções destes *workflows* bem como dos dados e parâmetros de entrada. A área que lida com estes dados, propondo técnicas e padrões de armazenamento bem como de coleta é chamada de proveniência (do inglês *Data Provenance*). Em Buneman, Khanna e Tan (2001) é encontrado um trabalho completo sobre proveniência de dados. Apesar de inicialmente parecer se tratar de uma tarefa simples, a coleta e armazenamento de dados de proveniência não é uma atividade trivial. Problemas de granularidade da informação, tamanho dos resultados intermediários, etc. se fazem presentes. Além disso, devemos associar metadados aos dados de proveniência armazenados, pois, somente assim conseguiremos compreender futuramente, por exemplo, a qual medida ou métrica um determinado número (que foi armazenado) está associado.

- **Modificações Dinâmicas dos Parâmetros**: é importante para o cientista que o mesmo possa modificar valores de parâmetros durante a execução do experimento caso o mesmo não esteja tomando os caminhos que eram esperados. Em experimentos científicos a imprevisibilidade é um fator que deve ser levado em consideração.

### 2.3.3 O ciclo de vida de um *workflow* científico

Em termos conceituais, o ciclo de vida de um *workflow* científico, em geral, costuma representar o ciclo de vida do experimento (em nosso caso, de descoberta de conhecimento em textos) que está sendo reproduzido. De acordo com Moreau et al. (2003), o ciclo de desenvolvimento de um *workflow* científico pode ser dividido em seis etapas (estas etapas são genéricas e se referenciam a qualquer experimento que possa ser mapeado como um *workflow* cientifico).

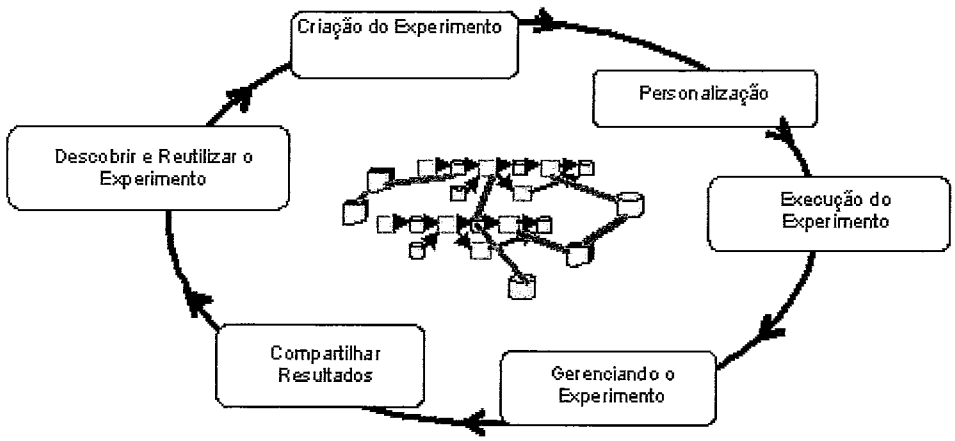


**Figura 2.2 - Ciclo de vida de um *workflow* científico, adaptado de (myGrid, 2008)**

Na figura 2.2 é possível observar o ciclo de vida de um experimento científico genérico proposto pela equipe do myGrid (2008). Esta figura pode nos mostrar como o ciclo de vida torna viável o processo de descoberta de experimentos científicos, envolvendo as informações sobre quem executou o experimento, quando, onde e por quê. Este processo de descoberta e reutilização está diretamente ligado com a padronização e sistematização do processo de definição do experimento, bem como de sua execução. Além disso, possibilita reutilizar experimentos que tenham sido executados anteriormente por outros membros de uma mesma equipe, por exemplo,

além de iniciar o desenvolvimento de um novo experimento. As etapas deste ciclo podem ser divididas em:

- Criação do Experimento: etapa na qual é gerado o *workflow* que representa o experimento que será representado. Visa à validação de uma hipótese através da invocação de serviços (*web* ou não). No caso de mineração de textos, esta etapa pode ser executada inúmeras vezes até que se alcance um resultado satisfatório.

- Personalização: neste passo devem-se registrar anotações que sejam relevantes para o experimento.

- Execução do Experimento: nesta fase ocorre a execução de cada uma das atividades do *workflow*, previamente definidas. Esta execução é realizada por uma máquina de *workflow*.

- Gerenciando o Experimento: aqui ocorrem verificações sobre o andamento do *workflow*.

- Compartilhar Resultados: nesta etapa ocorre a publicação do *workflow* gerado para que possa ser utilizado e avaliado em outros experimentos de outros analistas.

- Descobrir e Reutilizar o Experimento: é um refinamento da fase anterior, onde é possível encontrar um experimento previamente produzido e incorporá-lo a outro experimento.

### 2.3.4 O ciclo de mineração de textos visto como um *workflow* científico

Conforme foi explicitado na subseção 2.3.3 o ciclo de vida de um *workflow* científico possui diversas etapas em comum com os processos de descoberta de conhecimento (experimentos de mineração de textos) que são executados por um analista.

Assim como no ciclo de vida do *workflow* cientifico, o analista deve cumprir certas etapas no momento da composição do seu experimento. As etapas de definição e padronização obrigatoriamente já são executadas hoje (mesmo que sem apoio apropriado de alguma ferramenta), uma vez que o analista define quais atividades irão compor os seus experimentos. Anotações complementares normalmente são realizadas sobre essas escolhas de forma que se tenha uma documentação do processo (mesmo que ainda seja uma documentação mínima).

Uma vez definido, o experimento deve ser executado e controlado, a fim de que as atividades que compõem o fluxo recebam dados de entrada e geram as saídas esperadas. Principalmente no processo de descoberta de conhecimento em textos, o controle da execução se torna muito importante, pois muitas das implementações demandam um grande poder de processamento e muitas vezes muito tempo para terminarem suas execuções.

Finalmente com o resultado da execução em mãos, o analista pode verificar o quão correto e verossímil o resultado foi. Uma vez confirmado como válido o experimento, o analista deve então torná-lo disponível para outros analistas, a fim de que não ocorra re-trabalho, e seu experimento possa servir como base para novos experimentos no futuro.

Estas etapas, já cumpridas hoje por analistas de mineração de textos, coincidem exatamente com as etapas genéricas do ciclo de vida de um *workflow* cientifico, portanto é quase que intuitivo que visualizemos este processo como tal.

Além disto, Os *workflows* científicos podem ser considerados uma alternativa para modelar o processo de mineração de textos, uma vez que as etapas do processo de mineração de textos são muito bem definidas e independentes quanto à execução, e criam um ambiente satisfatório quando combinado com a tecnologia de serviços *web* (se mapearmos cada atividade do fluxo em um serviço *web* independente). Em Targino (2004) podemos encontrar um arcabouço voltado para *workflows* científicos para bioinformática. Apesar de este não ser o escopo deste trabalho, serve como referência para a criação de uma ferramenta que apóie o processe de criação, execução e redefinição destes *workflows* científicos.

É um fato conhecido na área de mineração de textos que o processo de mineração é sempre definido de acordo com o problema que se está tratando, por isso é altamente dinâmico. O fluxo de tarefas que será montado pelo analista de mineração de textos muda a cada novo problema, e, além disso, como nunca se sabe (*a priori*) a melhor solução para um determinado problema, várias tentativas devem ser realizadas até que se alcance um resultado satisfatório. Justamente por isso, os *workflows* para mineração de textos gerados podem sofrer constantes atualizações, seja por um refinamento desejado pelo analista ou para mudanças de etapas e tarefas que não são mais necessárias ao processo inicialmente modelado. O que se encaixa na definição de ciclo de um *workflow* científico, já que o processo de descoberta de conhecimento em textos é iterativo e interativo.

Conforme citado anteriormente nesta subseção, reutilizar dados gerados em experimentos (*workflows*) anteriores pode garantir uma economia de tempo e de poder de processamento. É muito interessante que o analista possa reaproveitar dados gerados por algumas tarefas em *workflows* criados e executados anteriormente, ou que possa reaproveitar informação gerada por *workflows* de terceiros, podendo algumas vezes reaproveitar o *workflow* por inteiro, se necessário. É igualmente interessante que o analista possa interromper a execução do seu *workflow* quando necessário e possa reiniciar o mesmo *a posteriori* do ponto onde foi interrompido, sem perdas. Para poder alcançar estes objetivos, se faz necessário que os resultados finais e intermediários (obtidos de cada tarefa do *workflow*) do processo sejam armazenados de forma adequada.

Informações complementares como anotações do analista podem ser anexadas aos dados armazenados de forma a dar semântica aos mesmos. Parte do significado dos dados está no processo que o gerou. Algumas perguntas do tipo "Como?", "Quando?" e "Por quê?" devem ser respondidas a fim de fornecer esse significado.

Segundo (Buneman, Khanna et al. 2001) a proveniência dos dados (ou *data provenance* do termo inglês), também denominada *pedigree* é a descrição da origem de um dado e de todo o processo pelo qual foi produzido. A proveniência se faz muito importante quando tratamos de qualidade dos dados, já que a mesma auxilia a formar uma visão da qualidade, da validade e de quão recente é a informação que está armazenada.

Ao visualizarmos o ciclo de vida de um processo de mineração de textos como um *workflow* científico, cada tarefa do processo será mapeada para uma ou mais tarefas do *workflow*, desta forma, podemos mapear o ciclo de vida de um experimento de mineração de textos em um *workflow* científico. A vantagem de se visualizar o ciclo de vida do processo de mineração como um *workflow* científico é que podemos utilizar, no âmbito da mineração de textos, as técnicas e ferramentas disponíveis para se trabalhar com *workflows* científicos, aproveitando-se assim de funcionalidades como coleta de dados de proveniência, redefinições, e etc, uma vez que esses assuntos já são um ramo de pesquisa bem difundido na área de *workflows* e várias soluções satisfatórias já foram propostas.

### 2.3.5 Sistemas Gerenciadores de *Workflows* (SGWf)

Uma definição importante é a de Sistema de Gerenciamento de *workflows* (oriundo do termo em inglês *Workflow Management System* – SGWf) definido em WFMC (2007). É o SGWf que tem como tarefa primordial fornecer o suporte computacional para que as atividades do desenvolvimento de um *workflow* possam ser executadas. Em outras palavras, é ele que tem como função invocar os serviços computacionais necessários (no nosso caso, serviços *web*) para a execução das atividades.

As atividades que o SGWf propicia são duas principalmente: a **especificação** de um *workflow* e a **execução** do *workflow* especificado anteriormente. A atividade de especificação pode ser definida como aquela em que o *workflow* é "desenhado", ou seja, a definição de qual forma ele irá ter. Como sub atividades da especificação do *workflow* podemos ter: quais atividades estarão presentes no *workflow*, qual o fluxo que a informação vai tomar, quais as entradas e quais as saídas vão existir, quais os meios de coordenação podem ser embutidos neste *workflow*, entre outras. De acordo com Targino (2004), um *workflow* W pode ser definido pela quádrupla (T,V,Sf,Cf), onde:

- T é um conjunto {t1, t2, . . . ,tn} de tarefas de W,
- V é um conjunto de variáveis {v1, v2, . . . ,vn} de W definindo um fluxo de dados,
- Sf é uma função sucessora associada a cada tarefa t ∈ , T, e

- Cf é uma função de condição associada a cada tarefa t ∈, T.

Enquanto que a atividade de execução do *workflow* é basicamente executar passo-a-passo as atividades definidas na etapa anterior, mantendo as restrições impostas na especificação.

Muitos SGWf consistem de uma máquina de *workflow* em um servidor de aplicações que interage com os clientes do *workflow* através de uma rede interna ou pela *internet*. A máquina de *workflow* controla e monitora de forma centralizada cada tarefa ou serviço, desde a instalação até a sua finalização. O banco de dados do *workflow* armazena informações sobre cada instanciação do *workflow* e o histórico detalhado das transações para monitoramento e documentação.

Alguns destes SGWf fazem uso de linguagens próprias para a definição do *workflow* (sendo essas linguagens de domínio público ou proprietárias), outros de diagramas, textos, documentos XML, etc. Vamos citar agora algumas das linguagens de definição de *workflows* existentes.

Podemos listar uma grande quantidade de linguagens para definir *workflows*. Nenhuma delas é homologada ainda como uma linguagem-padrão, por isso muitas continuam surgindo enquanto muitas não conseguem se estabelecer e acabam desaparecendo com o tempo. Dentre alguns exemplos encontrados na literatura podemos citar as linguagens BPEL4WS (Curbera, Golland e Andrews, 2003) (*Business Process Execution Language for Web Services*), XLANG (Thatte, 2001) (*Web Service for a Business Process Design*), WSFL (Leymann, 2001) (*Web Service Flow Language*), BPML (Arkin, 2001) (*Business Process Markup Language*), o SCUFL (SCUFL, 2007) (XML Simple Conceptual Unified Flow Language) e o MoML (MOML, 2007) (Lee e Neuendorffer, 2000).

Cada linguagem oferece recursos diferenciados, mas segundo comparação detalhada encontrada em Aalst e Hee (2003) e em Aalst, Hofstede e Kiepuszewski (2000), a maioria delas atende aos requisitos mínimos para a modelagem de um *workflow*.

Todas as linguagens oferecem ao menos meios de representação dos componentes básicos do *workflow* como atividades (ou serviços), fluxo de atividades, coordenação, etc.

Em particular, o BPEL4WS, o MoML e o SCUFL possuem um rico conjunto de construtores XML utilizados para coordenar processos de negócio encapsulados em serviços *web*.

A seguir dissertaremos sobre alguns dos SGWf que foram estudados, descrevendo vantagens e desvantagens de cada um, sempre mantendo o foco no processo de mineração de textos.

### 2.3.5.1 Taverna

Das diversas propostas de SGWf existentes e disponibilizadas para utilização, muitas delas foram desenvolvidas especificamente para atender as necessidades de um domínio de conhecimento específico. O Taverna é um exemplo deste tipo de SGWf, sendo focado principalmente nos requisitos de gerência de *workflows* para experimentos de bioinformática.

Desenvolvido na Grã-Bretanha através de um programa governamental de apoio ao desenvolvimento da bioinformática, o Taverna tem como objetivo facilitar o uso de *workflows* e computação distribuída pelos cientistas. É um projeto financiado pelo EPSRC (*Engineering and Physical Sciences Research Council*), envolvendo cinco universidades do Reino Unido, o grupo *European Bioinformatics* e alguns colaboradores da indústria privada.

É um *software* de código aberto orientado a serviços, implementado em Java e *scripts shell*, podendo ser utilizado em plataformas Windows ou Linux (o seu sítio contém instruções de instalação nos dois ambientes). Os *workflows* são definidos na linguagem Scufl através de uma interface gráfica intuitiva (Zhao, Goble e Stevens, 2004).

A Scufl (*Simple Conceptual Unified Flow Language*) é a linguagem padrão para definição de *workflows*, baseada em WSFL, e desenvolvida pela equipe do Taverna de acordo com as necessidades que o grupo tinha de possuir uma linguagem de definição o

mais simples possível. A linguagem deveria permitir ao cientista mapear tarefas conceituais em entidades simples com o mínimo de esforço.

O Taverna disponibiliza também uma série de serviços pré-definidos e estruturas de dados pertinentes ao domínio de bioinformática, além de suporte semântico através da utilização de ontologias e serviços de proveniência dos dados.

A utilização de ontologias dentro do Taverna tem como objetivo principal adicionar mais semântica a todo o processo e facilitar a tarefa de guiar o cientista no momento da definição do *workflow*, facilitando a descoberta de qual recurso utilizar no *workflow* que está sendo desenvolvido.

O sistema possui um repositório de definições dos *workflows*, dados do usuário e metadados e provê um mecanismo que armazena automaticamente os dados intermediários, que foram obtidos na execução, em um esquema relacional (proprietário) de proveniência.

Este processo de armazenamento é obtido através da sua própria máquina de execução (denominada no passado de FreeFluo). Entretanto, o Taverna não oferece suporte especializado para o processo mineração de textos (uma vez que foca no domínio da bioinformática), apesar de estar preparado para executar qualquer tipo de *workflow* científico que seja orientado a serviços (*web* ou não).

O projeto Taverna pode ser considerado um dos mais completos na literatura atual. Ele disponibiliza diversas funcionalidades para definição dos serviços, utiliza serviços *web* com possibilidade de utilização de serviços em grade (apesar de esta funcionalidade ainda não se encontrar 100% ativa). Entretanto, três fatores importantes fazem com que este projeto não possa ser utilizado no âmbito de mineração de textos:

- As ontologias que são definidas para o Taverna são focadas no domínio de bioinformática, não atendendo a alguns requisitos para oferecer suporte aos analistas de mineração de textos.

- Carecem de projetos mais divulgados que utilizem esta ferramenta para que se possa ter uma visão da aplicação destas tecnologias em um

ambiente real de produção que não seja um laboratório com pesquisas de bioinformática.

- O esquema de proveniência, bem como o processo de captura dos dados são proprietários e altamente dependentes da máquina de execução de *workflows* e do modelo de dados utilizado pelo Taverna, uma vez que ainda não há uma definição de um modelo padrão de proveniência.

### 2.3.5.2 Kepler

Diferentemente do Taverna, outros SGWf procuram não ser focados em apenas um domínio de conhecimento. O Kepler é um exemplo deste tipo de SGWf (Altintas et al, 2004).

O Kepler é um SGWf cuja máquina de execução de *workflows* é baseada na máquina de execução do antigo projeto Ptolemy II. O Kepler, assim como o Taverna, é um sistema de código aberto, porém com a grande diferença que o mesmo tem como objetivo atender múltiplos domínios do conhecimento, como a geologia, física, química e bioinformática, diferentemente do Taverna que era focado no domínio de bioinformática.

A versão atual do Kepler herdou uma série de características do projeto Ptolemy, porém é baseada em um modelo de classes diferente das do Ptolemy. Este modelo é capaz de representar e operar sobre inúmeros tipos de dados heterogêneos, sem contar a possibilidade de invocar serviços *web* e serviços em grade (o Kepler possui uma gama muito maior de componentes para serviços em grade do que o Taverna, tornando-se assim uma ferramenta muito atrativa visto o rápido crescimento e adoção das tecnologias de grade, como o Globus Toolkit (GLOBUS, 2007)).

O Kepler possui uma interface gráfica, denominada Vergil (Altintas et al, 2004), que proporciona o desenvolvimento rápido e satisfatório dos mais variados tipos de *workflows*, atendendo a grande parte dos requisitos propostos por Aalst, Hofstede e Kiepuszewski (2000).

Os objetos a serem incorporados ao *workflow* definido no Kepler se dividem em duas categorias: os atores (*actors*) e nos modelos de computação (*directors*). Os atores

podem ser definidos como os componentes que serão executados no *workflow*, enquanto que os diretores são encarregados da execução do e gerenciamento do ciclo de vida do *workflow* segundo um determinado modelo de computação.

O Kepler possui arquitetura modular baseada na tecnologia Java. A arquitetura oferece meios para permitir que usuários mais experientes e desenvolvedores de *software* possam implementar um número qualquer de modelos de computação. A arquitetura do Kepler é composta por um grande número de pacotes genéricos (pacotes que implementam os modelos de computação e atores, que já foram descritos anteriormente) e pacotes específicos. De acordo com Altintas et al (2004) a versão atual do Kepler foi construída baseada no arcabouço do PtolemyII e já oferece suporte total ao ciclo de vida de um *workflow* científico.

A interface do Vergil é bem intuitiva, e está divida em três áreas principais. A primeira área principal é a de menu de comandos e a barra de ferramentas, que permitem que o usuário execute, pare ou interrompa a execução de um *workflow*, ou ainda adicione portas de entrada e saída em um determinado ator composto. A segunda área é a paleta de componentes. Nesta paleta estão dispostos os atores e diretores que podem ser acoplados no *workflow* que está sendo desenvolvido.

O suporte à proveniência de dados é um requisito importante para os SGWf científicos, porém o Kepler ainda não oferece a essa característica de forma nativa. Para contornar essa limitação Altintas, Barney e Jaeger-Frank (2006) propuseram um tipo especial de diretor que será o responsável dentro do *workflow* por capturar e armazenar os dados de proveniência que forem gerados.

Atualmente, o componente que oferece suporte à proveniência é o Provenance Framework. (KPF, 2007). Ele foi proposto de forma a oferecer suporte aos projetos multidisciplinares do Kepler, porém nenhuma versão (nem mesmo preliminar) deste componente foi disponibilizada para testes e validações pela comunidade científica.

Apesar de possuir vantagens como o grande número de atores disponibilizados (mais de 160) ou a existência de atores específicos para acesso aos recursos de grade, o Kepler é pobre no suporte ao desenvolvimento de *workflows* no que se refere a guiar o usuário durante a elaboração do *workflow*. O Kepler não faz o uso de ontologias para facilitar o processo de desenvolvimento do *workflow* e nem para descrever dados e

objetos atinentes ao sistema. O arcabouço de proveniência, apesar de proposto, ainda não foi analisado e não pode ser considerado, *a priori*, como uma solução satisfatória para o requisito de proveniência de dados. Além destes fatores, devemos citar que o Kepler não dispõe de nenhuma ferramenta especializada no processo de mineração de textos (atores, por exemplo). Para se tornar uma ferramenta satisfatória no âmbito do processo de mineração de textos, o Kepler necessitaria ser estendido de forma que fosse mais aderente aos requisitos para modelar um *workflow* deste tipo.

#### 2.3.5.3 VisTrails

O VisTrails é uma opção ao Taverna e ao Kepler e tem se mostrado uma grande promessa dentre os SGWf. O VisTrails é um SGWf desenvolvido na universidade de Utah, nos Estados Unidos e que tem como principal meta prover suporte a exploração de dados e visualização. Inclusive a parte gráfica (visualização dos resultados) é um ponto em que o mesmo está bem avançado em relação aos demais SGWf.

Atualmente o VisTrails possui versões para os mais diversos sistemas operacionais como o Windows XP, Mac OS X e o Linux, com algumas pequenas variações na interface dependendo do sistema operacional adotado.

No VisTrails, o usuário está apto a criar e editar seus próprios *workflows* utilizando a interface gráfica do VisTrails chamada VisTrail Builder

O VisTrails Builder está dividido em três grandes áreas: uma lista de módulos (semelhantes aos atores do Kepler ou aos Processadores no Taverna), a área de design, onde o usuário montará o seu *workflow* e uma área especifica para exibir as propriedades dos módulos que forem adicionados ao *workflow*.

Uma grande vantagem do VisTrails é que todo o projeto foi concebido com foco em proveniência dos dados. O VisTrails foi o primeiro SGWf a armazenar a evolução de *workflows*, focando na proveniência da etapa de definição de *workflows*. Recentemente, foi desenvolvido um *plugin* para armazenar os dados de execução de *workflows* em uma base de dados local, sendo arquivos XML a solução padrão.

O VisTrails foi desenvolvido com a finalidade de prover mecanismos para a visualização múltipla de resultados de *workflows* através de uma infra-estrutura que pode ser utilizada com qualquer sistema de visualização existente.

O VisTrails possibilita ao usuário definir *workflows* e controlar suas alterações, ou seja, as diferentes versões do *workflow* que foi gerado e posteriormente modificado, podem ser visualizadas em modo visual, em árvores, de uma maneira simples e eficaz.

Desta maneira, é possível analisar de uma maneira mais simples e concluir quais parâmetros dados como entrada para um determinado experimento geraram o melhor resultado, simplesmente comparando visualmente as execuções destes *workflows* em uma janela especial dentro do VisTrails.

Assim como o Kepler, o VisTrails não possui suporte a ontologias e nem ferramentas específicas de suporte ao desenvolvimento do *workflow* ou para mineração de textos, apesar de prover um mecanismo de visualização poderoso que pode se mostrar muito útil no processo de descoberta do conhecimento em textos.

## 2.4 Considerações Finais

Após uma análise criteriosa tanto das ferramentas de mineração de texto quanto dos SGWf pudemos chegar à conclusão que nenhuma delas apóia satisfatoriamente o ciclo de vida de um experimento de mineração de textos. Consideramos fundamental o apoio não apenas ao *workflow* em execução, mas principalmente ao ciclo de vida do experimento. Os sistemas existentes não associam diversas execuções de *workflows* como fazendo parte de um único experimento, por exemplo. Neste ponto, o VisTrails talvez seja o SGWf que mais perto chega de atender este requisito, porém, não trabalha com o conceito de experimento cientifico.

Além disto, consideramos que o apoio de uma ontologia de domínio é essencial para o experimento desta natureza. Em nenhuma das ferramentas analisadas encontramos uma ontologia definida para mineração de textos, e, em nenhum dos SGWf analisados, o apoio à criação de uma ontologia e sua associação ao experimento. O Taverna/myGrid é talvez o SGWf que mais se aproxime de atender este requisito, porém, sua ontologia de domínio é totalmente voltada para experimentos em bioinformática.

As ferramentas específicas de mineração de textos oferecem os algoritmos e implementações das mais variadas e interfaces das mais completas do que se refere à facilidade e praticidade. Entretanto, nenhuma das analisadas possui mecanismos necessários já disponibilizados pelos SGWf e nem estão aderentes a padrões abertos como serviços *web*. Pode se tornar uma tarefa complicada gerenciar experimentos de mineração de texto sem dados de proveniência e sem um sistema que gerencie estes dados de maneira eficaz.

Em relação aos dados de proveniência, cada ferramenta possui mecanismos de gerência de dados integrados com esquemas de proveniências específicos para a ferramenta, e, cada um dos SGWf analisados possui um mecanismo distinto para armazenar dados de proveniência (os que o possuem), sendo que os mecanismos não são padronizados e nem compatíveis entre si. Além disto, os esquemas de proveniência são proprietários e, em alguns casos, inelegíveis através de consultas ao banco de dados.

Entretanto, cada um dos SGWf analisados possui inúmeras vantagens que podem ser devidamente aproveitadas no apoio ao ciclo de vida de um processo de mineração de textos. O VisTrails, por exemplo, possui uma preocupação grande com visualização e com o controle de versões dos *workflows* (a definição do *workflow* é considerada proveniência de dados). O Taverna, por sua vez, possui um mecanismo de proveniência e de tolerância a falhas que não é encontrado em mais nenhum SGWf disponível além do suporte semântico e dos mecanismos de proveniência. Já o Kepler possui uma gama de atores muito extensa (mais de 160) e componentes que disponibilizam acesso rápido a bases de dados e ao *grid*.

O que podemos concluir é que, atualmente, os SGWf analisados possuem diversas "falhas" ou requisitos que ainda devem ser atendidos. Entretanto, a tendência é que estas ferramentas se firmem como tecnologias utilizadas pela comunidade e superem estes problemas. Estas ferramentas vêm sofrendo evoluções constantes e a tendência é que sempre incluam novos mecanismos cada vez mais complexos e completos. Por exemplo, desde Outubro de 2005, o Taverna, já teve quatro versões liberadas (1.2, 1.3, 1.4 e 1.5), o que mostra que investimento está sendo realizado e que o projeto tem continuidade.

Desta forma, não precisaremos "re-inventar a roda" para oferecer apoio ao ciclo de vida de mineração de textos. Não parece fazer sentido que construamos um novo SGWf devido a algumas características especificas do domínio, mesmo porque estes SGWf possuem equipes enormes que os desenvolvem, e não dispomos desta estrutura. Com base nesta análise, concluímos que a solução mais interessante seria oferecer uma camada intermediária entre o usuário e o SGWf, oferecendo serviços que complementem o estado da arte dos SGWf.

Para comparar as propostas existentes, utilizamos como base as dez características de avaliação de SGWf (10+C) propostas por Targino (2004), acrescidas de algumas características pertinentes à avaliação de SGWf aderentes ao processo de mineração de textos. A seguir, é mostrada uma tabela (Tabela 2.1) comparativa entre os SGWf que foram analisados, indicando se os mesmos aderem (✓), aderem parcialmente (Ⓟ) ou não aderem (⊗) aos requisitos levantados.

Tabela 2.1 Avaliação dos SGWf de acordo com os critérios propostos em Targino (2004)

| *Características dos SGWf* | *Taverna* | *Kepler* | *VisTrails* |
|---|---|---|---|
| *Definição Abstrata do Workflow* | Ⓟ | ⊗ | ⊗ |
| *Workflow em termos de programas executáveis* | ✓ | ✓ | ✓ |
| *Geração Automática do Workflow* | ✓ | ✓ | ✓ |
| *Execução do Workflow* | ✓ | ✓ | ✓ |
| *Re-Execuções Parciais* | Ⓟ | Ⓟ | ⊗ |
| *Tratamento de Exceções* | ✓ | Ⓟ | ⊗ |
| *Registro de Execução dos Programas* | ✓ | ⊗ | ✓ |
| *Registro de Execução dos Workflows* | ✓ | ⊗ | ✓ |
| *Execução Remota* | ✓ | ✓ | ✓ |
| *Distribuição Através da Internet* | ⊗ | ⊗ | ⊗ |
| *Utilização de Ontologias* | ✓ | ⊗ | ⊗ |
| *Permite Intervenção Humana* | ✓ | ⊗ | ✓ |
| *Suporte Específico para mineração de textos* | ⊗ | ⊗ | ⊗ |
| *Utilização de serviços em grade* | Ⓟ | ✓ | ⊗ |
| *Mecanismo de Proveniência Nativo* | ✓ | ⊗ | Ⓟ |
| *Suporte Semântico Através de Ontologias* | ✓ | ⊗ | ⊗ |

# Capítulo 3 *MF-Ontology: Uma* Ontologia para Mineração de Textos

Nos últimos anos, a utilização de ontologias como modelo de representação de conhecimento aumentou em grande escala nas diversas áreas de conhecimento e pesquisa. Diversas propostas, das mais diferentes áreas, baseadas na utilização de ontologias surgem a cada dia. A utilização de ontologia como uma especificação conceitual compartilhada de um domínio (Gruber, 1993) foi um dos alicerces deste trabalho.

Este capítulo define conceitos básicos e apresenta a ontologia do domínio de mineração de textos que foi construída para dar apoio ao processo de definição de *workflows* e para classificar dados de proveniência recolhidos durante a execução dos *workflows*.

### 3.1.1 Ontologias como modelos conceituais de domínio

Existem inúmeros trabalhos na literatura que definem o conceito de ontologia. Segundo Gruber (1993) uma ontologia pode ser definida como "uma especificação explícita e formal de um conceito compartilhado". Noy e McGuiness (2001) definem uma ontologia como "uma descrição formal e explícita de um domínio de conhecimento". Estas duas definições combinadas (uma vez que podem ser consideradas complementares, já que apenas uma trata do compartilhamento e a outra fixa um domínio especifico de conhecimento) foram adotadas nesta dissertação, pois propomos uma ontologia como uma representação formal e explicita do domínio de mineração de textos (domínio específico) e, com isso, temos a intenção de compartilhar estes conceitos definidos dentro do ambiente proposto (*MiningFlow).* Outras definições podem ser encontradas na literatura, mas em geral, são variações das citadas anteriormente. Guarino (1998) ainda propõe a classificação de ontologias em quatro categorias distintas:

- Ontologias de Alto-Nível: descrevem conceitos mais gerais e abrangentes como ***espaço*** e ***tempo***, independentes de domínio.

- Ontologias de Tarefas: têm como objetivo representar uma tarefa em alto nível como, por exemplo, a compra de um automóvel.

- Ontologias de Domínio: descrevem um domínio específico de conhecimento. São por vezes denominadas "vocabulários controlados" sobre um domínio ou área de conhecimento, entretanto ontologias possuem elementos, como por exemplo relações ou propriedades, que as distinguem de apenas um vocabulário controlado.

- Ontologias de Aplicação: são uma "fusão" de uma ontologia de tarefa com uma ontologia de domínio, descrevendo conceitos que estão ligados diretamente a uma aplicação em um determinado domínio do conhecimento.

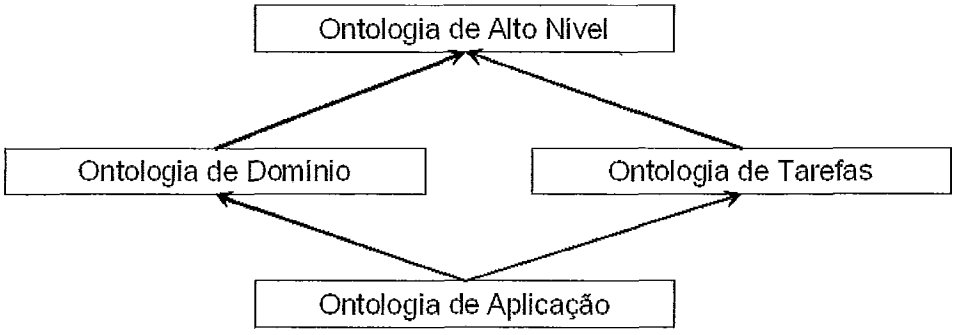


**Figura 3.1 Classificação dos diferentes tipos de ontologias. Setas representam relações de especialização, adaptado de Guarino (1998).**

Segundo Mastella et al. (2005) podem existir outras classificações de ontologias como as ontologias temporais ou ontologias de eventos que não são o foco deste trabalho.

Em ciência da computação, as ontologias são tipicamente voltadas para facilitar o compartilhamento e o reuso de informações, bem como incorporar semântica a processos de desenvolvimento de *software*, por exemplo.

As ontologias de domínio, foco principal deste capítulo, representam conhecimento declarativo sobre um domínio especifico: conhecimentos principais, seus atributos e relações entre eles, regras, axiomas, etc.

Por representarem um domínio específico de conhecimento, este tipo de ontologia vem sendo muito utilizada em pesquisas científicas, em especial no domínio da bioinformática. Iniciativas como o *Open Biological Ontologies* (OBO, 2008), um endereço *web* que contém uma série de ontologias de domínio que cobrem os mais diversos ramos da biologia. Estas ontologias vêm sendo utilizadas em massa nos sistemas de apoio a experimentos *in-silico,* de forma a oferecer mais semântica e a disponibilizar um vocabulário controlado e padrão para os experimentos em bioinformática.

### 3.1.2 Construtos de Ontologias de Domínio

Uma ontologia é composta de diversos "componentes" chamados de construtos. Gruber (1993) identificou cinco tipos de construtos ontológicos básicos: classes, relações, funções, instâncias e propriedades.

*Classes* representam elementos de um domínio de conhecimento e que são definidas por um conjunto de atributos, propriedades e seus possíveis valores.

*Relações* são construtos que representam um tipo de relacionamento entre dois construtos diferentes (ou entre o mesmo construto).

*Funções* são tipos especiais de relações que mapeiam um ou mais elementos do domínio para um único elemento e *axiomas* que são sentenças sobre o domínio que são sempre verdadeiras.

*Instâncias* são os indivíduos de uma ontologia.

*Propriedades*, *Slots* ou *Roles* são características de uma classe e as restrições de propriedades são chamadas de *facets* ou *role restriction.*

As descrições de conceitos em uma ontologia são efetuadas utilizando-se uma organização taxonômica, ou seja, baseada em generalização e especialização. Aspectos composicionais, isto é, relacionamentos do tipo "*parte-de/todo*" são ortogonais às

ontologias e devem ser representados através de funções não taxonômicas, isto é, propriedades.

### 3.1.3 Padrões de Representação de Ontologias

Para se tornarem artefatos concretos e assim poderem ser utilizadas por aplicações computacionais, as ontologias de domínio necessitam de uma linguagem de representação. Mas, assim como ocorre em outras áreas de pesquisa em ciência da computação, não existe um padrão rígido para representação de ontologias. Muitos padrões novos são propostos, porém a grande maioria deles se baseia em XML (W3C, 2006), criando desta forma seus próprios marcadores específicos (também chamados de *tags*). Entretanto, apenas alguns destes padrões são recomendados pelo W3C (W3C, 2007a).

Alguns exemplos de linguagem de descrição e construção de ontologias são o RDF ou *Resource Decription Framework Schema Language* (W3C, 2004), a OIL ou *Ontology Interchange Language* proposta por Fensel (2000) e sua extensão/adaptação a DAML+OIL ou DARPA *Agent Markup Language – Ontology Interchange Language* proposta por Horrocks et al. (2002). A OWL ou *Web Ontology Language* (W3C, 2007d) é uma proposta de linguagem de construção de ontologias baseada nas linguagens OIL e DAML+OIL. Estas linguagens (de nível lógico) de representação de ontologias foram cuidadosamente projetadas para prover tanto expressividade quanto eficiência computacional (GUIZZARDI, 2005). A OWL facilita a possibilidade de interpretação por máquinas de conteúdo da *web* do que o XML, RDF e RDFS uma vez que fornece vocabulário adicional com uma semântica formal. Por ser definida sobre a linguagem XML, a OWL viabiliza a interoperabilidade entre aplicações construídas sobre diferentes sistemas operacionais e linguagens de programação. Além disso, é recomendada pelo W3C, o que a torna um padrão de fato.

A OWL foi a nossa escolha de linguagem de representação de ontologias para este trabalho, uma vez que é recomendada pelo W3C e possui uma série de aplicativos disponíveis para manipulação, incluindo o mais completo dentre os analisados, como será explicitado na subseção seguinte.

### 3.1.4 Ferramentas para Edição de Ontologias

Assim como nas propostas de linguagens de definição de ontologias, podemos encontrar uma gama muito grande de ferramentas disponíveis para trabalhar com estas linguagens. A seguir exibimos um resumo das principais ferramentas encontradas na literatura e que foram testadas e justificamos nossa escolha pelo Protégé (PROTEGE, 2007).

O *Web*Onto (Domingue, Motta e Corcho, 1999) é uma ferramenta que possibilita a navegação, criação e edição de ontologias de forma simples e eficiente. Possui uma ferramenta gráfica para criação da ontologia e verificação de consistência da herança. Disponibiliza uma biblioteca com mais de cem ontologias de exemplo que podem ser consultadas. Entretanto, esta ferramenta trabalha apenas com a linguagem OCML (Medeiros et al., 2007), que não é um padrão atual, o que impactou negativamente na escolha desta ferramenta para a nossa modelagem.

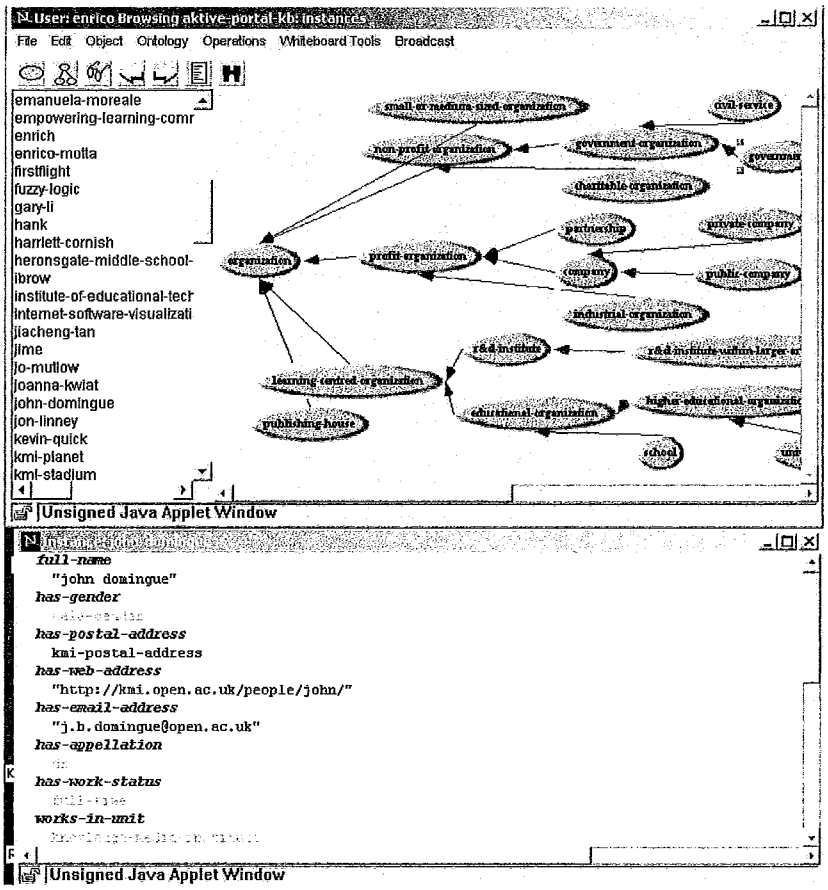


**Figura 3.2 Interface do *Web*Onto.**

O OilEd (Bechhofer et al., 2001) é um editor de código aberto que permite a construção de ontologias na linguagem OIL. Porém, esta aplicação não é um ambiente completo para construção de ontologias uma vez que não disponibiliza verificação de consistência.

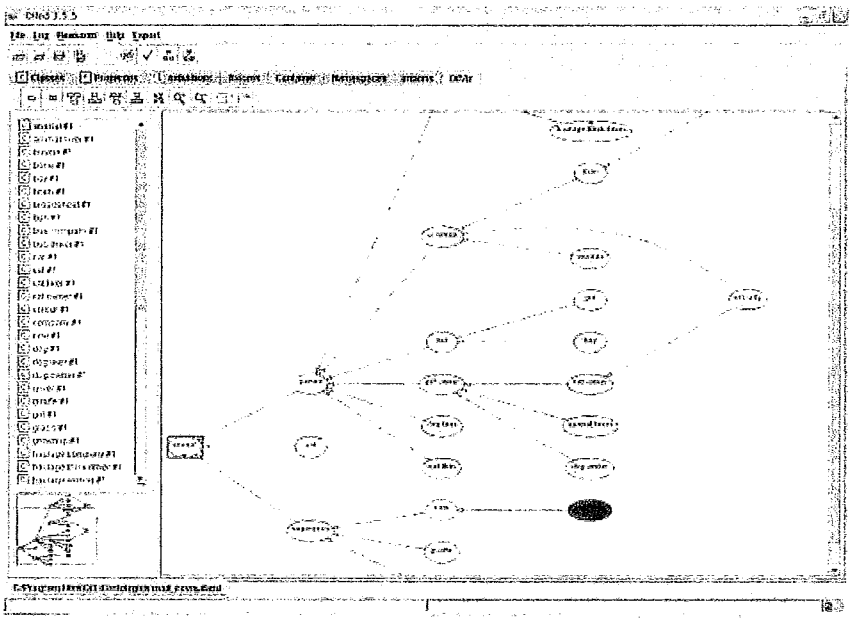

**Figura 3.3 Interface do OilEd.**

O OntoEdit (Maedche et al., 2002) é um ambiente gráfico para representação, inspeção, navegação e testes de ontologias. As ontologias são armazenadas em bancos relacionais e podem ser implementadas em XML, RDF e DAML+OIL.

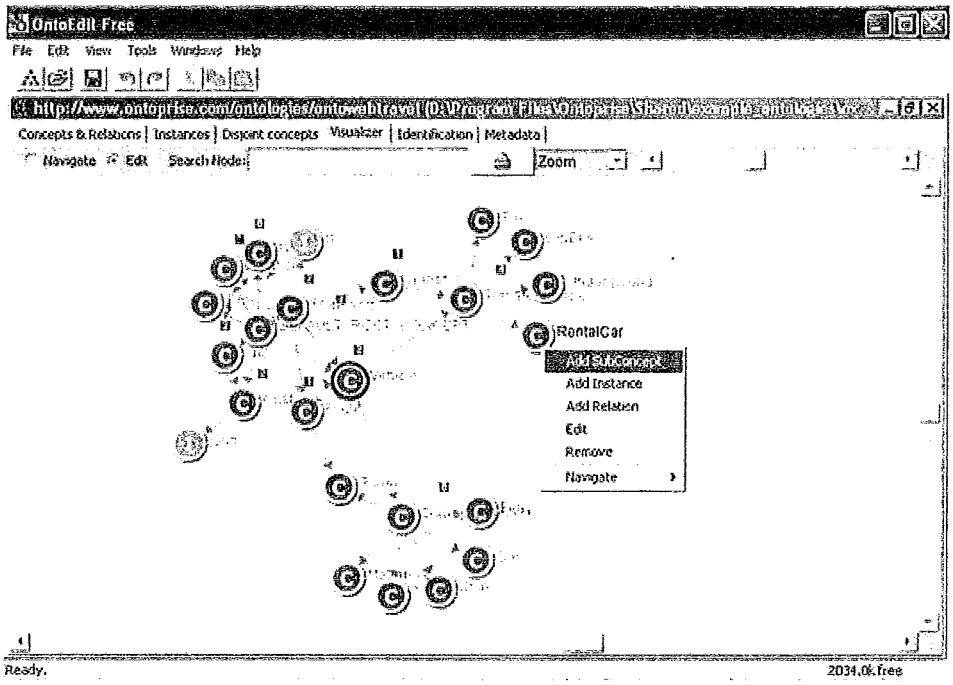


**Figura 3.4 Interface do OntoEdit.**

O Protégé (PROTEGE, 2007) é um ambiente interativo de código aberto que proporciona uma gama de funcionalidades para criação, edição e manipulação de ontologias de maneira gráfica. A arquitetura do Protégé é modular e possibilita a inserção de novos recursos e funcionalidades por parte do usuário. O Protégé tem se firmado como uma ferramenta amplamente utilizada, com uma comunidade fixa de usuários e desenvolvedores com cerca de cem mil pessoas. Devido à quantidade de funcionalidades disponibilizadas, entre elas um *plug-in* para OWL, o Protégé foi a nossa escolha de editor para a modelagem e o manuseio da ontologia de mineração de textos.

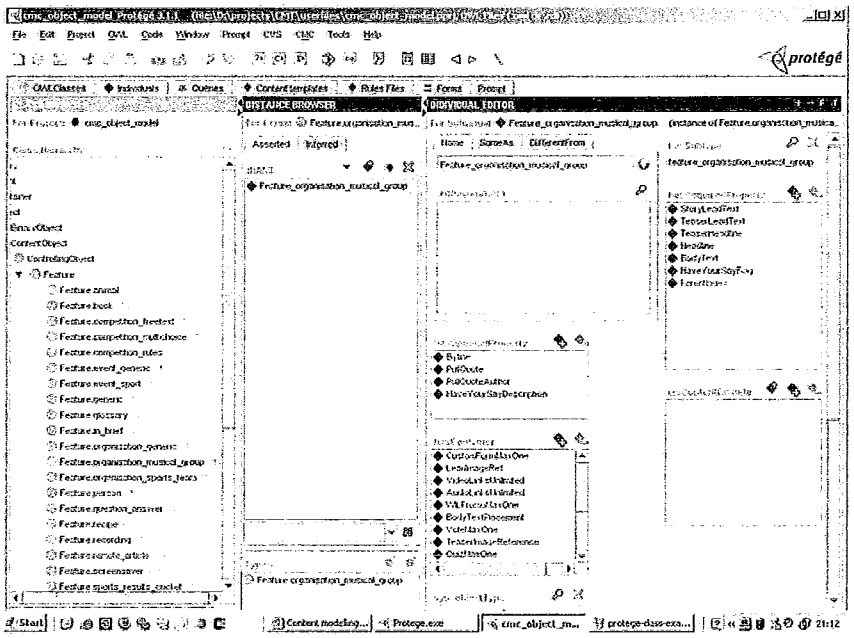

**Figura 3.5 Tela do Protégé.**

## 3.2 Ontologias em Mineração de Dados

Em toda literatura existem algumas propostas de ontologias voltadas para o processo de mineração de dados (CANNATARO e VELTRI,2006)(CANNATARO e COMITO, 2005). Entretanto, não foi encontrada na literatura nenhuma ontologia que descrevesse o processo de mineração de textos.

Nesta subseção exibiremos e dissertaremos sobre algumas das ontologias para mineração de dados que foram encontradas na literatura e que serviram de base para a construção da ontologia para mineração de textos que iremos propor a seguir nesta dissertação.

### *3.2.1 WekaOntology*

Esta ontologia para mineração de dados for proposta por Cannataro e Veltri (2006) para ser utilizada na ferramenta MS-Analyzer (*Mass Spectra Analyzer*) proposta no mesmo trabalho.

O MS-Analyzer é uma aplicação que visa analisar massa espectral, ou seja, uma grande massa de dados de medidas (intensidade, por exemplo) que representam a

abundância de biomoléculas que possuem certa massa. Esta aplicação possui um editor de *workflows* baseado em ontologias que utiliza uma ontologia chamada *WekaOntology*.

A *WekaOntology* é uma ontologia que modela as ferramentas de mineração de dados do pacote Weka (WEKA,2007) enriquecida com a descrição de conceitos de algoritmos de pré-processamento de massa espectral (finalidade principal da aplicação) e da *ProtOntology*, que modela conceitos, métodos, algoritmos, ferramentas, e bancos de dados relevantes para o domínio proteômico e provê uma base biológica para que se possa fazer uma análise dos dados futuramente. Infelizmente esta ontologia não se encontra disponível para consulta, o que tornou inviável que a utilizássemos como base para o trabalho ou como referência para o mesmo.

### 3.2.2 DAMON (*Data Mining Ontology*)

Esta ontologia de mineração de dados foi proposta por Cannataro e Comito (2003) como uma solução para oferecer suporte semântico ao usuário no momento da definição de seu *workflow* de mineração de dados dentro do *Knowledge Grid* (Cannataro e Talia, 2003b).

Este trabalho propõe uma ontologia para o as tarefas de mineração de dados, não englobando as fases de pré-processamento e nem de pós-processamento. O artigo propõe a modelagem do processo de mineração de dados em torno de quatro conceitos-chave:

1. **Tarefa**: explicita e modela as tarefas de mineração de dados existentes.
2. **Método**: modela os métodos de mineração de dados que existem (por exemplo: árvore de decisão).
3. **Algoritmo**: modela as propostas de algoritmos que são baseados em métodos.
4. ***Software***: modela as implementações de determinados algoritmos.

Utilizamos a DAMON como base para a ontologia proposta para mineração de textos uma vez que a mesma se encontrava disponível para análise e modelava de maneira genérica um processo de mineração de dados, sem incorporar características

particulares de nenhum domínio (como é o caso da *WekaOntology* que incorpora conceitos de massa espectral na ontologia).

## 3.3 *MF-Ontology*: Uma Ontologia de Mineração de Textos

### 3.3.1 Objetivo

A ontologia que propomos nesta dissertação, denominada *MiningFlow Ontology* ou simplesmente *MF-Ontology*, tem como objetivo criar uma representação conceitual, formal e compartilhada do domínio do processo de mineração de textos.

No ciclo de vida de um experimento de mineração de textos,a fase de pré-processamento se faz importante, uma vez que visa homogeneizar os dados de entrada em apenas um padrão definido. Esta fase não é modelada em nenhuma das ontologias de mineração de dados que se encontravam disponíveis para análise, logo necessitaríamos construir uma ontologia nova ou estender alguma já existente.

A *MF-Ontology* visa principalmente modelar o ciclo de vida de um experimento de mineração de textos, mas também visa atender algumas das seguintes necessidades dos usuários da ferramenta que utilizará esta ontologia para apoiar o usuário na definição de seu experimento. Estas necessidades serão abordadas na seção 3.3.2 deste capítulo.

### 3.3.2 Construção da Ontologia

A construção da ontologia mencionada (*MF-Ontology*) foi feita totalmente de forma manual. Os conceitos foram criados e as instâncias inseridas através de consultas a bibliografia consagrada na área de mineração de textos, em especial a Feldman e Sanger (2006) e aos especialistas na área de mineração de textos. Desta forma, não foi utilizada nenhuma ferramenta de extração de conceitos para inserirmos os dados na ontologia, apenas a ferramenta Protégé (PROTEGE, 2007) para modelagem e inserção dos dados (criação das instâncias).

Durante a etapa de criação, utilizamos como base a metodologia para criação de ontologias proposta por Noy e McGuiness (2001) (que consiste em perguntas e atividades a serem executadas durante a construção de uma ontologia) para estender e

adaptar a ontologia DAMON proposta por Cannataro e Comito (2003), gerando uma nova ontologia que atendesse os requisitos e cobrisse o domínio de mineração de textos.

O **primeiro passo** para a construção da ontologia foi a definição do domínio que a ontologia esta proposta a cobrir: em nosso caso o domínio é a mineração de textos. Além do domínio, necessitamos saber **quais tipos de questões** nós devemos ser capazes de responder com a ontologia:

1. **Identificar processos equivalentes**: mapeando o ciclo de vida através de conceitos da ontologia, podemos identificar quais processos definidos são equivalentes a outros, apenas verificando quais se referenciam a um mesmo conceito na ontologia. Por exemplo: se uma atividade está designada para executar uma função de pré-processamento dentro do *workflow* e o usuário não sabe quais serviços escolher para executar esta atividade, a ontologia deve ser capaz de informar quais serviços estão disponíveis para executar a atividade que foi determinada pelo conceito associado.

2. **Identificar tipos de dados**: através da ontologia devemos poder identificar quais os tipos de dados de entrada e saída de cada processo (seja ele um serviço, programa local, serviço em grade, etc.), fazendo com que possamos executar uma checagem de tipo de dados na definição do nosso experimento, evitando-se assim erros mais comuns. Por exemplo: se uma atividade de pré-processamento gera um arquivo texto como saída e a entrada do serviço seguinte lê os dados em forma de planilha, estes serviços são incompatíveis e a ontologia deve ser capaz de analisar este tipo de problema.

3. **Identificar a seqüência de atividades dentro do experimento**: a ontologia deve ser capaz de identificar se uma atividade depende de outra para poder ser definida no experimento. Por exemplo: se o usuário define que uma atividade de mineração de textos será executada antes da atividade de pré-processamento, a ontologia deve ser capaz de verificar que a ordem de execução das atividades não faz sentido.

4. **Relacionar os dados de proveniência com os conceitos**: através do uso da ontologia para apoiar o ciclo de vida de mineração de textos, podemos associar também seus conceitos aos dados que forem armazenados após as execuções dos experimentos pelos SGWf. Desta forma, o dado armazenado passa a possuir semântica.

O **segundo passo** da metodologia é definir quais são os conceitos chave e as propriedades da ontologia. Utilizamos um processo de construção *top-down*, onde mapeamos os conceitos mais genéricos da ontologia e depois os especializamos.

Baseando-se na definição do processo de descoberta de conhecimento em textos (ou simplesmente KDT, to termo em inglês *knowledge Discovery on texts*), extraímos e definimos os principais conceitos-chave: *Passo do KDT, Tarefa, Função, Algoritmo, Método, Medida, Software* e *Dado.*

Um *Passo do KDT* representa as macro-atividades do ciclo de vida de mineração de textos, e está subdividida em: *Pré-Processamento, Mineração de Textos e Pós-Processamento.*

Uma *Tarefa* representa uma técnica de mineração de textos que é utilizada para extrair padrões e modelos de um conjunto de dados não estruturados. Em outras palavras, as tarefas de mineração de texto são o objetivo principal do ciclo de vida de um experimento de mineração de textos.

Uma *Função* representa uma técnica de preparo dos dados para a atividade de mineração ou uma técnica para avaliação dos modelos gerados. Assim sendo as funções foram subdivididas em *Funções de Pré-Processamento* e *Funções de Pós-Processamento* de acordo com o seu objetivo.

Um *Algoritmo* representa o modo como uma *Tarefa* ou uma *Função* é executada.

Um *Método* representa a metodologia utilizada como base por um algoritmo para descobrir conhecimento. Diferentes métodos servem para diferentes propósitos.

Uma *Medida* é uma representação de como um algoritmo de mineração de texto se comportará. Um mesmo algoritmo de mineração de texto pode ser executado com

diversas medidas de distância diferentes, por exemplo. A escolha da medida para o algoritmo poderá causar impacto no resultado final obtido.

Um *Software* é a implementação de um determinado *Algoritmo*. Este é um conceito importante para a proposta desta dissertação, uma vez que um dos usos da ontologia será guiar o usuário no processo de definição de seu *workflow* de KDT. O ambiente computacional (SGWf) em que esta definição se dará manipula, essencialmente, seqüências lógicas de ***softwares* de mineração de texto**. Neste contexto, é imprescindível que a MFOntology possa guiar o ambiente a responder as seguintes questões: Que tarefa de mineração de textos, ou função de pré/pós-processamento, um determinado *software* executa? Que metodologia um *software* utiliza? Que tipos de dados um *software* manipula? Que algoritmo o *software* implementa?

Um *Dado* é a representação semântica de uma saída ou entrada de qualquer *Software*.

Uma vez que tenhamos definido os conceitos-base da ontologia, devemos definir quais são os relacionamentos e propriedades destes conceitos. Por exemplo: a propriedade *Disponível* de um *software* que indica se o mesmo está disponível para uso ou não. Outros exemplos que podemos citar são: o relacionamento *Executa Tarefa* que relaciona os conceitos *Algoritmo* e *Tarefa de Mineração de Texto*, o relacionamento *Precede* que indica qual *Passo do KDT* precede o outro em uma ordem lógica. Por exemplo, na ontologia proposta, *Pré-Processamento* deve sempre preceder *Mineração de Textos*, ou ainda o relacionamento que define que um *Método* SEMPRE especifica uma *Tarefa* ou uma *Função*.

O **terceiro passo** da construção da ontologia define a hierarquia de conceitos através de taxonomias. Taxonomias são usadas para organizar conceitos ontológicos e utilizam dois tipos de relacionamentos:

1. "*é-um*" (do inglês *"is-a"*): generaliza ou especializa conceitos na ontologia. Por exemplo: *Classificação* "é uma" *Tarefa de Mineração de Textos*.

2. "*parte-de*" (do inglês *"part-of"*): define partições de uma classe. Por exemplo: *Text Mining* "é parte dos" *Passos de KDT*.

A ontologia foi então construída utilizando diversas pequenas taxonomias especializadas e inter-relacionadas. Exibimos a seguir algumas das taxonomias (parte delas) que foram definidas na construção da ontologia para mineração de textos. O arquivo de definição da ontologia construída completa, com todas as suas taxonomias, é apresentada no Anexo C desta dissertação. As linhas tracejadas representam os relacionamentos "*é-um*" e "*parte-de*" entre as classes e as linhas contínuas as instâncias.


**Figura 3.6 Taxonomia de Passos do KDT.**

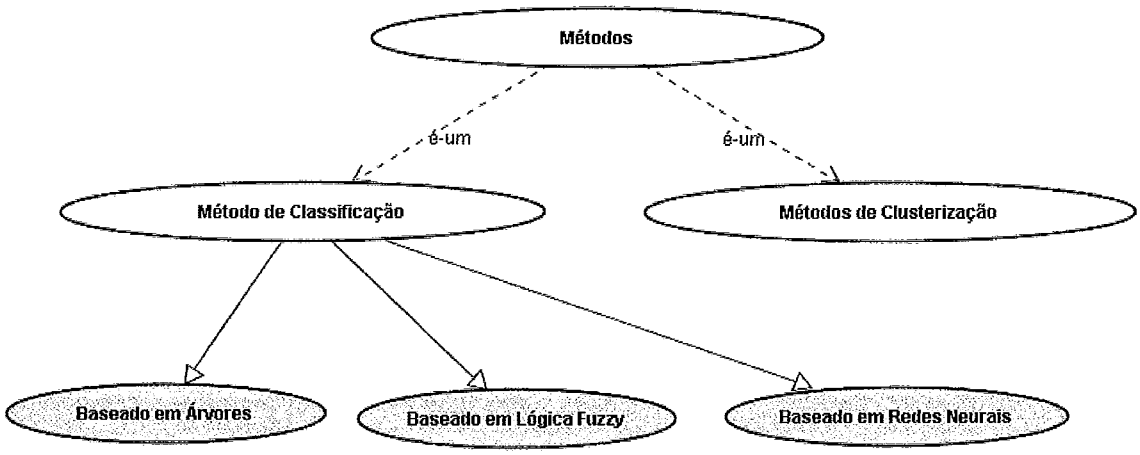


**Figura 3.7 Taxonomia de Métodos do KDT.**

O **quarto passo** da construção da ontologia é definir os axiomas (afirmações que sempre são verdadeiras) para relacionar as taxonomias que foram geradas para os conceitos de *Passos do KDT, Método, Algoritmo, Software* e *Dado.*

No domínio de mineração de texto, podemos usar axiomas para representar afirmações como as abaixo representadas:

- Um *Software* SEMPRE implementa pelo menos um *Algoritmo.*
- Um *Algoritmo* SEMPRE utiliza pelo menos um *Método.*
- Se um *Software* implementa um *Algoritmo*, e este *Software* é de classificação, o mesmo deve implementar um *Algoritmo de Classificação* (restrição de integridade da ontologia).
- Um *Método* SEMPRE especifica uma *Tarefa* ou uma *Função.*

Desta forma, definimos as relações entre as taxonomias, gerando uma rede de conceitos que agora pode ser utilizada para atender às necessidades anteriormente levantadas. A Figura 3.8 nos mostra parte da ontologia alcançada. Nesta figura exibimos parte da ontologia referente apenas ao *Software de Classificação* fictício X, que é baseado no *Algoritmo* $X_1$ , que por sua vez utiliza o *Método* baseado em árvores. Uma observação importante deve ser feita sobra a Figura 3.8: o relacionamento *executa* deveria estar representado entre as classes *Algoritmo de Classificação* e *Método de*

*Classificação*, entretanto, para evitar a poluição da imagem e uma visualização ruim da figura, este relacionamento foi mantido apenas entre as instâncias.

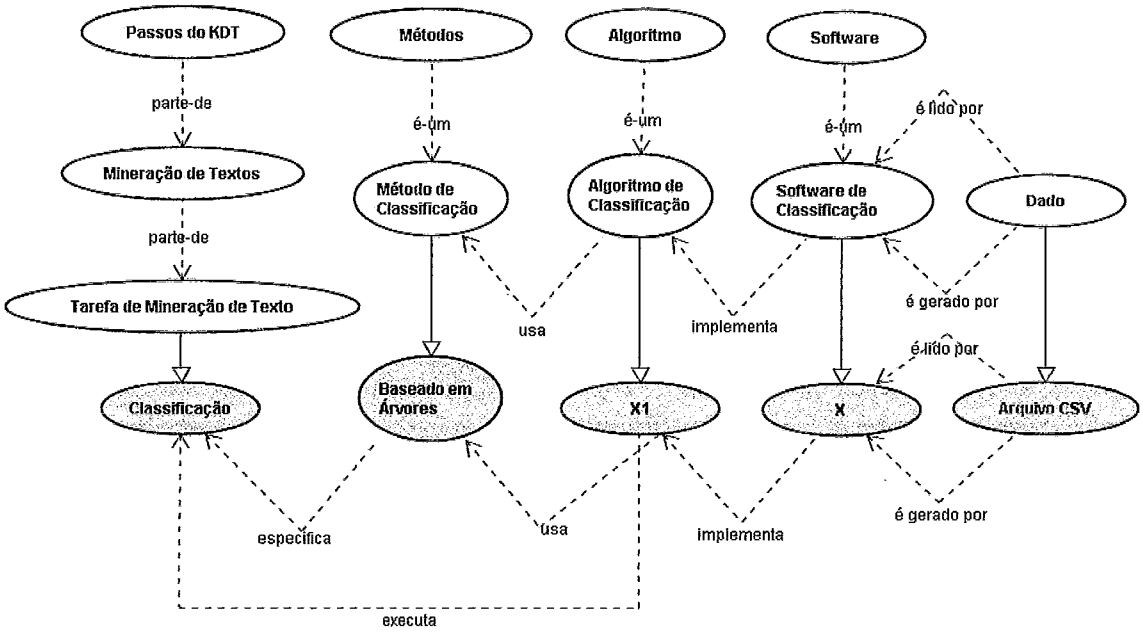


**Figura 3.8 Parte da *MF-Ontology*.**

É importante ressaltar que a *MF-Ontology*, no seu estágio atual, contempla apenas o sub-domínio de algoritmos, métodos e *softwares* de mineração de textos que estão aplicados no espaço geométrico. Outros sub-domínios de mineração de textos, por exemplo em que os algoritmos ou experimentos são baseados no espaço de conjuntos não estão cobertos.

### 3.3.3 Implementação em OWL

A partir do modelo da ontologia MF-Ontology, utilizamos a ferramenta Protégé (PROTEGE, 2007) para gerar um artefato concreto do modelo desenvolvido. Com o editor gráfico da ferramenta, que implementa todos os construtos necessários à nossa modelagem, tivemos apenas que recriar os conceitos existentes na ontologia utilizando a sintaxe da ferramenta. A ferramenta se encarregou de converter nossa modelagem para a linguagem OWL. Nesta subseção, exibimos um fragmento do arquivo OWL gerado que define o conceito de *Passo do KDT* e suas subclasses.

```
<owl:Class rdf:ID="Step_KDT">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Step KDT</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Step_PreProcessing">
        <rdfs:subClassOf rdf:resource="#Step_KDT"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Step PreProcessing</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Step_Visualization">
        <rdfs:subClassOf rdf:resource="#Step_KDT"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Step Visualization</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Step_Text_Mining">
        <rdfs:subClassOf rdf:resource="#Step_KDT"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Step Text Mining</rdfs:label>
</owl:Class>
```

**Figura 3.9 Parte da definição OWL da *MF-Ontology*.**

## 3.4 Validação da Ontologia

### 3.4.1 Metodologia de Validação

O processo de modelagem e construção de uma ontologia é, apesar dos muitos esforços da comunidade cientifica, quase que um trabalho artesanal. É uma afirmação bem conhecida pela comunidade científica que para um mesmo domínio podem existir diversas ontologias diferentes, cada qual com sua particularidade na modelagem. Justamente devido a esta variedade, é uma tarefa extremamente complexa avaliar uma ontologia e definir se a mesma está "certa" ou "errada" uma vez que um mesmo conceito pode ser modelado corretamente de formas diferentes.

Na literatura, podemos encontrar algumas propostas para validação de ontologias. Uma das mais proeminentes metodologias propostas é a *OntoClean*, proposta inicialmente por Guarino e Welty (2002). Tipicamente, esta metodologia tem servido de prova da usabilidade de uma ontologia que foi desenvolvida.

Neste trabalho, a validação da MF-Ontology foi realizada utilizando-se tanto a proposta da OntoClean quanto a validação através de especialistas.

### 3.4.2 Validação da MF-Ontology através da metodologia OntoClean

A *OntoClean* é baseada em noções filosóficas, focando em "limpar" taxonomias que compõem a ontologia. A metodologia se baseia em quatro definições fundamentais: *Rigidez (Rigidity), Identidade (Identity), Unidade (Unity) e Essência (Essence).* Associando estas definições como meta-propriedades dos conceitos que compõem uma taxonomia, podemos fazer com que estas meta-propriedades representem o comportamento dos conceitos. Estas meta-propriedades impõem restrições (Guarino e Welty, 2000) que são utilizadas para validar uma ontologia, de acordo com o *OntoClean.*

#### 3.4.2.1 Essência e Rigidez

A primeira noção formal que o método discute é a *Essência.* Cada propriedade de uma classe pode ser considerada essencial se e somente se esta propriedade for verdadeira para cada instância da classe. Por exemplo, a propriedade "***ser duro***" é essencial para a classe "***Martelo***" em uma ontologia, mas não é essencial para a classe "***Esponja***", por exemplo. Existem ainda propriedades que nunca são essenciais (*Anti-Essenciais*).

O Conceito de *Rigidez* é uma forma especial de *Essência*: uma propriedade é rígida se e somente se ela é essencial para todas as instâncias das classes. Tomemos como exemplo uma classe "***Pessoa***", se existisse uma propriedade "***Ser Pessoa***" ela seria essencial e rígida. Já no caso anterior, "***Ser duro***" pode ser essencial para a classe martelo, mas não rígido, pois no caso das esponjas ela não é essencial. A propriedade "***Ser Duro***" é chamada de *Semi-Rígida,* pois é essencial para uma classe e não essencial para outra. Toda propriedade da ontologia deve ser classificada em *Rígida, Semi-Rígida* e *Anti-Rígida.*

Desta forma, para garantir que a ontologia esteja correta os autores propõem algumas restrições para validar a mesma como, por exemplo: "Nenhuma propriedade *Anti-Rígida* pode classificar propriedades *Rígidas*".

### 3.4.2.2 Identidade e Unidade

Segundo Guarino e Welty (2002), embora sejam propriedades sutis, *Identidade* e *Unidade* são as meta-propriedades mais importantes para a metodologia *OntoClean*. Embora sejam consideradas definições diferentes, *Identidade* e *Unidade* estão diretamente relacionadas.

*Identidade* refere-se ao problema de ser capaz de reconhecer todas as entidades como sendo a mesma ou como sendo diferentes e *Unidade* se refere a estar apto a reconhecer todas as partes que formam uma entidade.

Tomemos a seguinte situação como exemplo. Duas classes chamadas "***Duração***" (com valores "uma hora", "duas horas") e "***Intervalo de Tempo***" (com valores 1:00-2:00 24/12/2007, por exemplo). A proposta é colocar "***Intervalo de Tempo***" como subclasse de "***Duração***". Neste caso, a correção vem da análise da meta-propriedade *Identidade*: as diversas instâncias de "Uma hora" que podem ser criadas em "***Duração***" são idênticas. Entretanto dois intervalos que acontecem ao mesmo tempo são iguais, porém em horários distintos, são diferentes, mesmo que o tamanho do intervalo seja igual. Isto cria uma contradição: se todas as instancias de "***Intervalo de Tempo***" são instancias de "***Duração***" (uma vez que é uma subclasse), como duas instâncias são geradas de uma classe cuja classe-pai só pode gerar uma instância? Nesse caso ao invés de assumir que "todos os intervalos são durações de tempo", deveríamos ter escrito "todos os intervalos possuem durações de tempo" e colocar duração como uma nova propriedade e não como uma classe na ontologia.

O Conceito de *Unidade* é mais sutil: imaginemos uma classe chamada "***Água***", onde suas instâncias seriam quantidades de água. Como não há um senso de "todo" e "parte", essa entidade não poderia existir.

Já uma entidade "***Oceano***" poderia ter como instâncias "Oceano Atlântico", "Oceano Pacífico", "Oceano Índico", etc. O que nos dá a noção de todo e parte. Desta forma, não podemos modelar na ontologia a classe "Oceano" como subclasse de

"***Água***", uma vez que a classe "***Oceano***" nos da idéia de todo e parte e a classe "***Água***" não.

A proposta da metodologia *OntoClean* é caracterizar as propriedades e buscar inconsistências baseando-se em restrições como estas que foram mostradas (lista não exaustiva das restrições).

### 3.4.2.3 Implementação da Metodologia

Para utilizar esta metodologia, se faz necessário que a mesma esteja implementada em uma das ferramentas de modelagem e construção de ontologias já apresentada. A *OntoClean* já possui uma implementação ponta para o Protégé em forma de *plug-in*. Através desta implementação, podemos validar as meta-propriedades na *MF-Ontology* no próprio Protégé, estando aptos a verificar se a ontologia está "limpa" e não viola nenhuma regra proposta pela metodologia. Foi implementada uma taxonomia no próprio Protégé para a classificação de ontologias. Através desta taxonomia, o Protégé consegue executar as restrições e verificar a correção da ontologia. A figura 3.10 nos mostra a taxonomia definida para que possamos utilizar a metodologia *OntoClean* no Protégé.

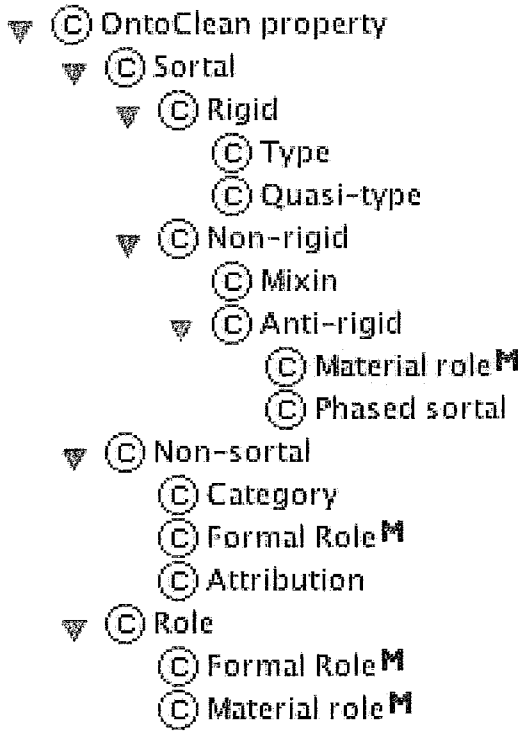


**Figura 3.10 A Taxonomia de Meta-propriedades proposta na metodologia *OntoClean*.**

### 3.4.3 Validação da Ontologia MF-Ontology com Especialistas

Apesar da metodologia OntoClean ser considerada uma referência entre as técnicas de avaliação de ontologias, a mesma não possui recursos para avaliar uma ontologia em todos os aspectos. Por ser uma metodologia independente de domínio, a OntoClean não se propõe a avaliar se uma ontologia é adequada para um determinado domínio ou área de conhecimento, ou seja, se algum conceito do domínio não foi modelado, por exemplo. A técnica utilizada para validar a ontologia em relação a estes problemas foi validar a modelagem através de entrevistas com especialistas no domínio de mineração de textos.

Estas entrevistas seguiram o roteiro presente no anexo E desta dissertação. Este roteiro foi elaborado de maneira que as perguntas feitas pudessem traduzir e validar alguma meta-propriedade utilizada pelo *OntoClean* e desta forma pudéssemos encontrar problemas antes mesmo de submetermos a *MF-Ontology* ao método. Além disso, este questionário nos deixaria aptos a verificar se algum conceito não foi previamente modelado e desta forma pudéssemos incorporá-lo na ontologia.

Tomemos como exemplo duas classes da ontologia, A e B, onde B é subclasse de A. Contextualizando na *MF-Ontology* podemos assumir que A seja a classe *Método* e B a classe *Método Baseado em Árvores.* Uma pergunta presente no questionário de avaliação é a seguinte: "É possível que um método baseado em árvores não seja um método?" ou ainda "O método baseado em árvores é SEMPRE um método de mineração de textos?". Com estas perguntas estamos validando os axiomas da ontologia e caso a resposta seja ***Não*** para o primeiro caso, e ***Sim*** para o segundo, já teríamos validado a meta-propriedade *Identidade.* No formulário de avaliação foram criadas perguntas para validar as classes, propriedades e relacionamentos (construtos da ontologia), sempre relacionando uma pergunta com uma meta-propriedade que está sendo avaliada.

### 3.4.4 Resultados Obtidos com as Validações

As validações foram realizadas, como explicitado anteriormente, em duas fases distintas: validação com especialistas e a aplicação do método *OntoClean* na *MF-Ontology.*.

Na etapa de validação da ontologia com especialistas foram entrevistados três especialistas do domínio de Mineração de Textos, todos com qualificada formação e experiência acadêmicas. São professores universitários, com diversas publicações nacionais e internacionais em conferências e revistas acadêmicas.

Nesta fase da validação não foram detectados erros nas definições dos conceitos existentes, mas sim a ausência de alguns conceitos, por exemplo:

1. Incluir a função de **lematização** na fase de pré-processamento: esta função pode ser considerada equivalente a *stemmização* quando o usuário trabalha com frases e não com palavras do documento. Neste contexto, remodelamos a ontologia, incluindo uma nova função de lematização dentre as funções de pré-processamento. Além disto, uma restrição teve de ser adicionada à ontologia, para representar funções "incompatíveis". Este caso, por exemplo, não faz sentido que um analista trabalhe com ***lematizaçao*** e ***stemmização***, assim sendo, eles se tornam incompatíveis em um mesmo experimento.

2. Incluir **Associação, *Link Analysis* e Sumarização** nas tarefas de mineração de textos: estas tarefas são consideradas extremamente comuns no desenvolvimento de aplicações de mineração de textos e não poderiam ser desconsideradas.

3. Incluir **avaliação de modelos e comparação de resultados** como funções de pós-processamento.

Outras observações foram feitas em relação ao conceito de método da ontologia. Os métodos de mineração de textos são essencialmente derivados dos métodos de mineração de dados. Desta forma, todos os métodos (árvores de decisão, redes neurais, etc.) são também métodos do processo de mineração de dados. Entretanto, para a

validação que estamos nos propondo a realizar, este conceito pode ser considerado corretamente modelado uma vez que de acordo com as respostas pudemos garantir as meta-propriedades de *Identidade* e *Unidade,* necessárias para a validação posterior via *plug-in.*

O formulário completo obtido com as entrevistas pode ser consultado no anexo E desta dissertação.

A segunda fase de validações foi aplicar o método *OntoClean* na ontologia proposta. Através do *plug-in* do Protégé, cada classe, relacionamento e propriedade da *MF-Ontology* foi analisada pelo *plug-in,* gerando um relatório com as inconsistências.

A principal inconsistência encontrada pelo Protégé foi com relação à classe *Dado.* Inicialmente havíamos modelado duas subclasses de *Dado*: *Dado de Entrada* e *Dado de Saída.* Entretanto, estas duas classes não foram validadas em relação à meta-propriedade *Identidade* já que ambas possuíam as mesmas instâncias ("Documento PDF", por exemplo). Analisando a inconsistência, podemos perceber que um tipo de dado é o mesmo independente se é um dado de entrada ou um dado de saída. A mudança na modelagem foi retirar as subclasses e criar instâncias diretamente a partir da classe *Dado.* Para diferenciar um dado de entrada de um dado de saída, foram criadas duas propriedades em cada instância de *Software*: *Tipo de Entrada* e *Tipo de Saída* que referenciam as instâncias de *Dado.*

Outra inconsistência encontrada foi em relação à propriedade *WSDL* das instâncias relativas às classes de *Software.* Esta propriedade nem sempre será válida para todos os *softwares* (já que a ontologia foi modelada para qualquer tipo de *Software*), mas como a implementação atual do *MiningFlow* será baseada em serviços *web*, resolvemos manter esta propriedade mesmo "ferindo" a validação completa da ontologia.

Na versão atual da ontologia foram modeladas 49 classes, 13 relacionamentos entre as classes (e conseqüentemente entre as instâncias das mesmas), 21 propriedades de classes e instâncias, e para a versão do estudo de caso, a ontologia foi populada com 63 instâncias das classes definidas.

## 3.5 Conclusão

Para auxiliar o processo de definição de um *workflow* científico para mineração de textos, esta seção propôs uma ontologia de domínio construída (adaptada e estendida) a partir de uma ontologia de domínio mais genérica voltada para o processo de mineração de dados e, por isso, facilmente extensível a outros contextos do mesmo domínio (mineração de textos, *web mining*, etc.).

Esta ontologia pode ser facilmente incorporada ao SGWf de forma que possa guiar o usuário na construção de seu *workflow* de mineração de textos. A definição e construção da ontologia, além de fornecer subsídios para a construção da ferramenta *MiningFlow*, nos auxiliou muito na compreensão do processo de mineração de textos e a levantar as necessidades dos futuros usuários do sistema.

# Capítulo 4 A Arquitetura do *MiningFlow*

Conforme foi explicitado na introdução desta dissertação, o *MiningFlow* tem como objetivo principal criar uma camada intermediária entre o usuário final e o SGWf de forma a oferecer apoio ao ciclo de vida de um experimento de mineração de textos. O principio básico da proposta é combinar as tecnologias de *workflows* com o uso de uma ontologia de domínio (já explicada no capítulo 3 desta dissertação) de forma a criar um ambiente que possa "guiar" o analista durante seus experimentos. A proposta deve ser capaz de auxiliar o usuário tanto nas tarefas de definição do *workflow* quanto na execução do mesmo. Este capítulo tem como finalidade detalhar a arquitetura proposta como um todo, seus componentes, suas funcionalidades, seu modelo de dados e suas limitações.

Na seção 4.1 será discutida a arquitetura do *MiningFlow* e seus sete principais módulos: o módulo de registro de serviços, o módulo de definição abstrata do *workflow*, o módulo de definição concreta do *workflow*, o módulo de redefinição, o módulo de tradução, módulo de registro de proveniência e o módulo de consulta aos dados de proveniência. Na seção 4.2 são vistos os tipos de usuários que interagem com este ambiente e na seção 4.3 como estes usuários interagem durante o ciclo de vida de um experimento de mineração de textos.

## 4.1 Arquitetura do *MiningFlow*

A arquitetura do *MiningFlow* pode ser visualizada na figura 4.1, e possui ao todo sete módulos principais:

- Módulo de registro de serviços.
- Módulo de definição abstrata do *workflow*.
- Módulo de definição concreta do *workflow*.
- Módulo de redefinição do *workflow* concreto.
- Módulo de tradução.

- Módulo de registro de proveniência.
- Módulo de consulta aos dados de proveniência.

A arquitetura *MiningFlow* é dividida em três áreas principais, a saber:

- A **Camada de Aplicação**, onde se situam os componentes que efetivamente terão uma interação com o usuário final e que invocarão os módulos da arquitetura.
- A **Camada Núcleo** que agrupa os componentes chamados de núcleo do sistema. Estes componentes não são visuais ao usuário final do ambiente, porém desempenham um papel fundamental na composição do arcabouço.
- E por fim a **Camada Externa**, composta por serviços *web* de terceiros, e pela máquina de *workflow* que será utilizada para executar os experimentos definidos com o auxílio do *MiningFlow*.

Na figura 4.1, cada componente foi segregado por cores: os componentes azuis são os **componentes núcleo**, os componentes amarelos são os da **camada externa**, os componentes brancos os da **camada de aplicação** e o componente verde a nossa **base de proveniência**. Nas subseções que se seguem cada módulo da **camada núcleo** será detalhado de forma que o leitor possa ter um entendimento mais avançado da proposta desta arquitetura.

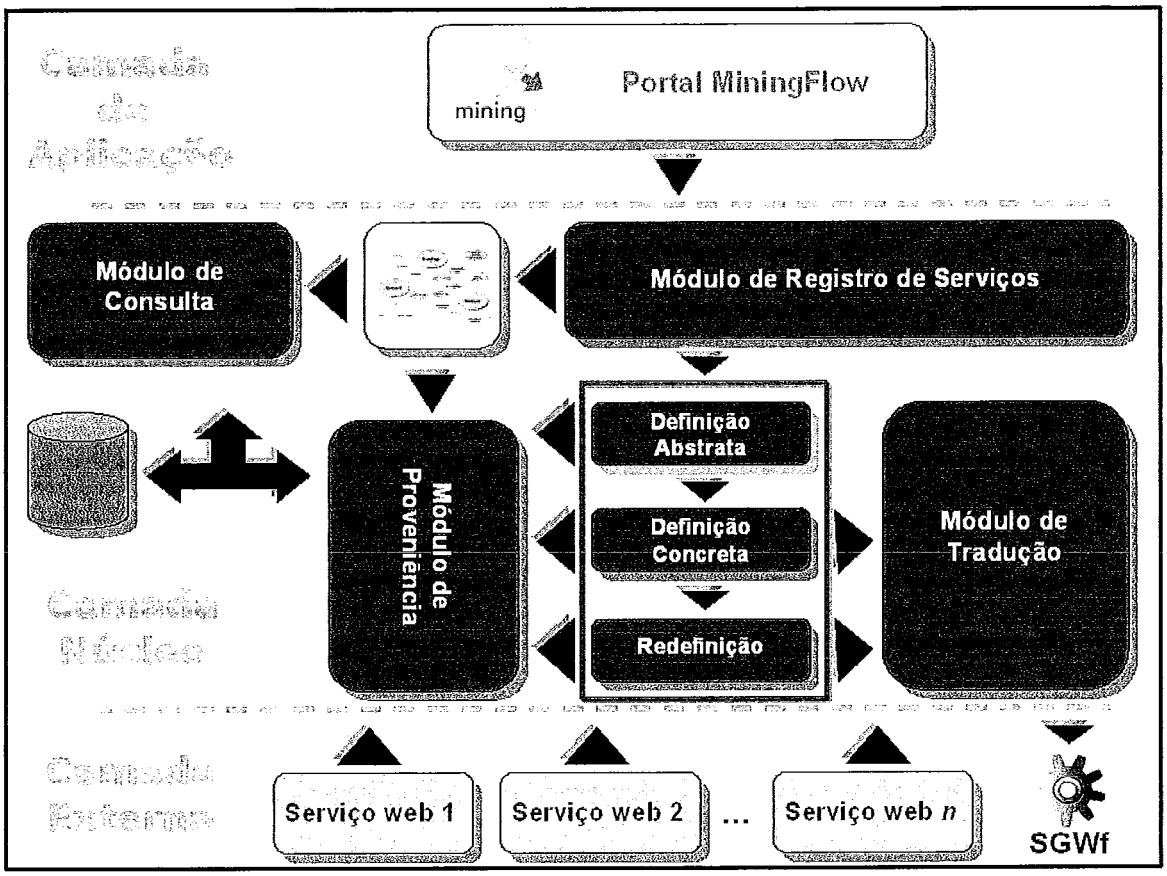


**Figura 4.1 Arquitetura do *MiningFlow*.**

### 4.1.1 Módulo de Registro de Serviços

Um dos objetivos principais da abordagem *MiningFlow* é criar um ambiente que apóie o analista na elaboração e execução de seu experimento de mineração de textos. Para isso, o *MiningFlow* combina a utilização de tecnologias de *workflows* com ontologias.

O módulo de registro de serviços é o componente responsável por integrar um SGWf a uma ontologia do domínio de Mineração de Textos, incorporando ao ambiente *MiningFlow* todas as definições presentes na ontologia. Este componente tem como finalidade principal identificar os elementos (as Classes, os Relacionamentos e as Instâncias) da ontologia selecionada e disponibilizá-los para os demais módulos do ambiente para consulta.

Apesar de ser sugerida nesta dissertação a utilização da *MF-Ontology* como a ontologia padrão para o *MiningFlow*, qualquer ontologia pode ser incorporada à

ferramenta, de forma que seus construtos possam ser utilizados no restante do processo por todos os outros componentes da arquitetura.

Este módulo é chamado de módulo de registro de serviços, pois partimos da premissa que a ontologia de mineração de textos que venha a ser utilizada tenha modelado o conceito de *software* de forma que qualquer programa ou serviço seja incorporado e disponibilizado para a ferramenta ao se efetuar a "carga" dos conceitos e instâncias da ontologia no ambiente *MiningFlow*.

### 4.1.2 Módulo de definição abstrata do *workflow*

Por não haver um consenso para os termos abstrato e concreto em *workflows*, esta dissertação adota as definições de (CAVALCANTI, 2003). Um *workflow* abstrato é uma definição de alto nível das atividades que compõem o *workflow* em questão. Esta definição se dá através de conceitos, ou no caso desta dissertação, de classes ontológicas. Um *workflow* concreto, por outro lado, é uma definição de mais baixo nível das atividades que compõem o *workflow*, onde estas atividades são diretamente associadas e identificadas pelos programas ou serviços que efetivamente serão executados (CAVALCANTI, 2003).

No ambiente *MiningFlow*, cada atividade do *workflow* será mapeada para uma classe da ontologia que estiver sendo utilizada para definir o *workflow* abstrato. Se o usuário optar por utilizar a *MF-Ontology* em sua definição, ele estará apto a escolher qualquer classe que defina uma etapa do processo de mineração de texto para associar a uma atividade do *workflow* abstrato que está sendo definido. É importante ressaltar que a arquitetura do *MiningFlow* foi projetada para ser independente da modelagem de ontologia. Assim sendo, cada ontologia pode ser modelada de maneira diferente e o usuário deverá selecionar quais classes da ontologia escolhida serão utilizadas para auxiliar na criação do *workflow* abstrato. A figura 4.2 nos mostra um exemplo de uma definição abstrata de um *workflow* seguindo as classes ontológicas propostas na *MF-Ontology*.

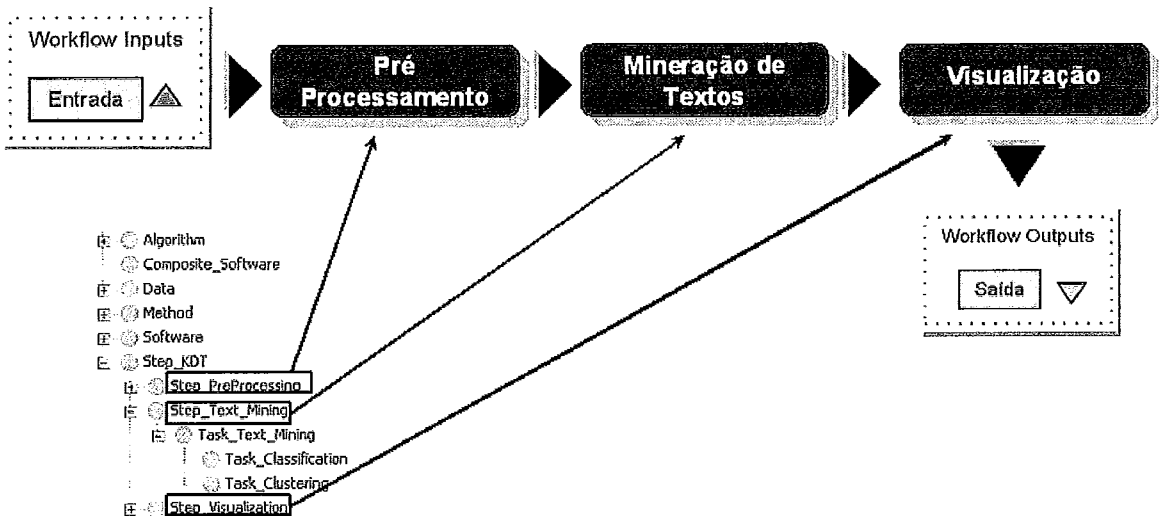


**Figura 4.2 Mapeamento das atividades do *workflow* abstrato em termos de classes ontológicas.**

No exemplo da figura 4.2 definimos um processo "clássico" de mineração de texto composto pelas três etapas principais propostas por Feldman e Sanger (2006): **pré-processamento, mineração de textos e pós-processamento (visualização)**.

Na ontologia escolhida, foram definidas classes que representam os conceitos etapa de pré-processamento (*Step_PreProcessing*), etapa de mineração (*Step_Text_Mining*) e etapa de pós-processamento *(Step_Visualization*). Estes conceitos contemplam, entre outras informações semânticas, uma definição para cada atividade do *workflow*. Após esta associação, o usuário tem capacidade de saber quais os conceitos que cada atividade executará, sem que a atividade esteja associada a um programa executável ou serviço.

Desta forma conseguimos incorporar semântica ao processo e à definição do *workflow*, uma vez que cada conceito escolhido terá uma definição explícita na ontologia que poderá ser consultada a qualquer momento.

É importante ressaltar que, no exemplo dado, foram utilizadas as classes mais "genéricas" da ontologia para se mapear as atividades do *workflow* abstrato. O usuário estará livre para escolher quaisquer classes da ontologia desejar. As atividades do *workflow* podem ser definidas de forma mais genérica (como no exemplo) ou o quão especializado se queira.

Por exemplo, o usuário pode desejar não incluir uma atividade que execute a etapa de pré-processamento por completo. Ao invés disso, ele deseja incluir atividades em seu *workflow* que executem as funções de *stemmização*, limpeza e remoção de *stop words* somente. O módulo estará apto a mapear estas atividades uma vez que estes conceitos estejam presentes na ontologia.

Outra vantagem que o módulo de definição de *workflows* abstratos disponibiliza, por utilizar *workflows* abstratos, é a pré-validação de um *workflow* que está sendo definido. Tomemos o exemplo anterior: conceitualmente a etapa de mineração de textos deve ser posterior à etapa de pré-processamento, para que os algoritmos da etapa de mineração de textos possam lidar com dados previamente trabalhados. Imagine que um usuário tenha incluído uma atividade no seu *workflow* abstrato referente à etapa de pré-processamento e logo após tenha definido uma atividade subseqüente associada ao conceito de mineração de textos. Na ontologia, estes conceitos possuem uma "precedência" definida, ou seja, existe uma propriedade ontológica que obriga que uma etapa de pré-processamento deve sempre preceder uma etapa de mineração de textos. Assim sendo, o módulo de definição do *workflow* abstrato deve ser capaz de identificar este tipo de restrição e impedir o usuário de definir um *workflow* conceitualmente errado.

Uma vez definido, o *workflow* abstrato tem sua definição armazenada em uma estrutura interna do sistema e o módulo de definição abstrata envia esta mesma definição para o módulo de registro de proveniência do sistema (o funcionamento do módulo de proveniência será explicado mais adiante).

### 4.1.3 Módulo de definição concreta do *workflow*

Seguindo o fluxo do sistema, encontramos o módulo de definição concreta do *workflow*. Este módulo tem como responsabilidade definir o *workflow* que será executado de fato pela máquina de *workflow* (*SGWf*). Este módulo importa uma definição abstrata de um *workflow*, interpretando-a de forma que possa auxiliar o usuário na escolha dos programas executáveis ou serviços que comporão seu *workflow*.

Uma vez carregada, a definição abstrata do *workflow* traz consigo que classes foram escolhidas para designar cada atividade do *workflow*. A ontologia *MFOntology* relaciona o conceito de *software* com a etapa ou sub-etapa do processo de mineração de

texto que o mesmo executa. Assim sendo, o módulo de definição concreta do *workflow* pode utilizar as classes ontológicas escolhidas na definição abstrata para sugerir quais programas ou serviços efetivamente executarão cada atividade do experimento. Além disto, o módulo de definição do *workflow* concreto deve ser capaz de auxiliar o usuário na redução de erros durante a definição de quais parâmetros serão dados como entrada para o *workflow* ou para cada um dos seus serviços.

A figura 4.3 nos mostra um exemplo simplificado do processo de geração do *workflow* concreto a partir de uma definição abstrata.

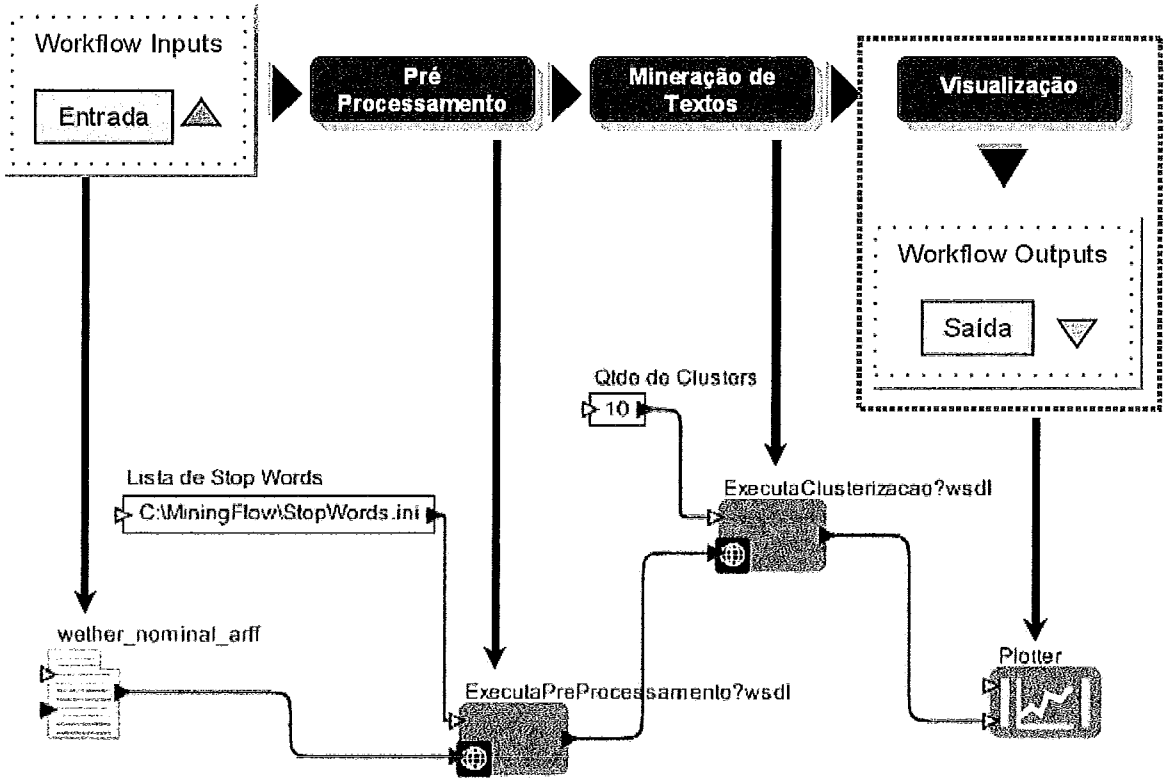


**Figura 4.3 Mapeamento das atividades do *workflow* concreto a partir das classes ontológicas definidas no *workflow* abstrato.**

A figura 4.3 ilustra que cada atividade do *workflow* abstrato foi mapeada para um serviço *web* que implementa o conceito explicitado pela atividade abstrata. Como exemplo, considere a classe abstrata pré-processamento. Esta classe abstrata está associada ao conceito *Step_PreProcessing* da ontologia. Suponha que exista na ontologia uma instância da classe *Software* cadastrada e que essa instância esteja associada ao conceito *Step_PreProcessing* através de um relacionamento que explicite

que este *software* executa determinado passo dentro do processo de mineração de textos. No momento da definição do *workflow* concreto, serão disponibilizados apenas os serviços cujas instâncias na ontologia executem o passo do processo de mineração de textos definido pelo conceito da classe abstrata do *workflow*. Voltando ao exemplo, no momento da definição concreta da primeira atividade do *workflow*, somente os *softwares* que estiverem associados à etapa de pré-processamento serão liberados para compor o *workflow* concreto.

Desta forma, conseguimos utilizar uma definição abstrata do *workflow* para gerar uma definição do mesmo em termos de programas executáveis ou serviços, ou seja, sua definição concreta.

Outra funcionalidade que é executada pelo módulo de definição de *workflows* concretos é a validação de tipos de entrada e saída de cada programa ou serviço *web*. Ao se escolher um *software* para compor a seqüência de atividades, o módulo faz a verificação da compatibilidade entre a saída de um serviço e a entrada do serviço subseqüente. Por exemplo: se a saída do serviço de pré-processamento da figura 4.3 informar apenas o caminho da rede onde a coleção pré-processada se encontra e o serviço de agrupamento (*clustering*) esperar como entrada os documentos pré-processados (não apenas o caminho, mas os documentos de fato), o *workflow* gerará um erro na sua execução (apesar de não gerar *a priori* um erro de definição). Para evitar problemas futuros, o módulo de definição de *workflows* concretos faz a checagem de tipo de acordo com o tipo de dado de entrada e tipo de dado de saída que foi informado para cada serviço na ontologia.

Uma vez definido, o *workflow* concreto tem sua definição armazenada em uma estrutura interna do sistema e o módulo de definição concreta envia esta mesma definição para o módulo de proveniência do sistema e para o módulo de tradução, que serão explicados com mais detalhes nas seções 4.1.5 e 4.1.6.

### 4.1.4 Módulo de redefinição do *workflow*

Este módulo tem como finalidade principal disponibilizar ao usuário a chance de alterar parâmetros de entrada do *workflow* concreto sem alterar seu formato. Experimentos onde o analista de mineração de textos utiliza a técnica chamada de validação cruzada (HAN e CAMBER, 2001), que propõe que se execute um mesmo

experimento *n* vezes, apenas variando os parâmetros de certas atividades, são um exemplo freqüente de oportunidade de uso deste módulo.

Desta forma, o usuário pode utilizar a definição concreta de um *workflow* específico e modificar apenas os valores de entrada do *workflow*. Na figura 4.4 exibimos um exemplo de redefinição do *workflow* concreto apresentado na subseção 4.1.3.

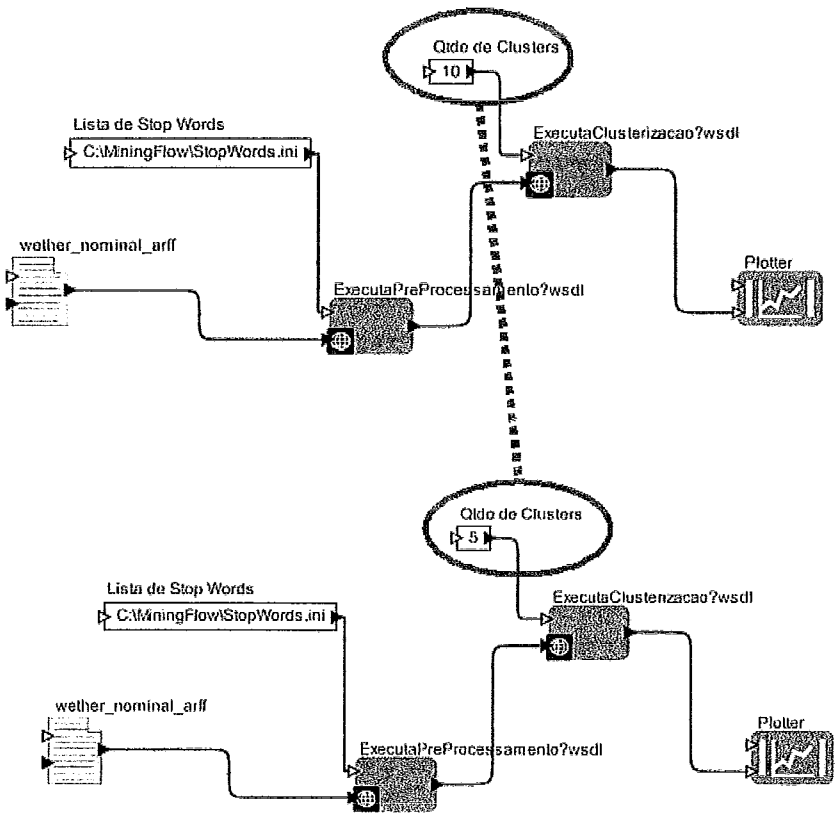


**Figura 4.4 Redefinição do *workflow* concreto através da modificação dos parâmetros de entrada.**

É importante ressaltar que a forma do *workflow* concreto, ou seja, quais serviços, a ordem dos mesmos ou os parâmetros de entrada existentes, não foram modificados. O módulo permite que apenas os valores dos parâmetros de entrada sejam alterados pelo usuário.

Uma vez redefinido, o *workflow* concreto tem sua redefinição armazenada em uma estrutura interna do sistema (esta definição permanece associada ao *workflow*

concreto "original") e o módulo de redefinição envia esta mesma redefinição para o módulo de proveniência do sistema e para o módulo de tradução.

### 4.1.5 Módulo de tradução

O modulo de tradução do *MiningFlow* é o responsável por gerar a definição do *workflow* concreto em uma linguagem de definição de *workflows* específica. Este módulo recebe como entrada a representação do *workflow* concreto na estrutura interna de classes do programa e uma opção de linguagem de definição de *workflows* a ser traduzida. O módulo então converte cada estrutura interna do sistema em *tags* XML que possam ser compreendidas pelos SGWf.

É importante ressaltar que para cada SGWf que seja incorporado à arquitetura do *MiningFlow*, deverá ser acoplada uma extensão ao módulo de tradução para que o mesmo realize o mapeamento adequado e gere uma saída que possa ser processada pelo SGWf escolhido.

Assim sendo, o módulo de tradução representa uma vantagem da arquitetura, já que o mesmo possibilita que o *MiningFlow* seja uma proposta independente de SGWf. Como já foi mencionado em capítulos anteriores, não existe um SGWf padrão e nem uma linguagem de definição padrão, logo é interessante para um pesquisador ou uma instituição que não se atenha a determinadas tecnologias que não sejam um padrão reconhecido ou um padrão de fato. Cabe destacar que a incorporação de uma opção de linguagem nova (linguagem de definição de um novo SGWf) ao *MiningFlow* é uma atividade de programação bem compartimentada e de extensão simples ao código.

### 4.1.6 Módulo de registro de proveniência

O módulo de registro de proveniência é o responsável por armazenar no banco de proveniência as informações pertinentes ao experimento em questão. Foi definido um esquema de proveniência para o *MiningFlow* (figura 4.5) que será utilizado por este módulo. Este esquema lógico de proveniência pode ser implementado em qualquer SGBD e tem como objetivo armazenar os dados que sejam importantes para que o experimento de mineração de textos possa ser válido.

Conforme já foi definido nos capítulos anteriores, o processo de armazenagem dos dados de proveniência é extremamente importante, pois somente através dela

podemos mostrar a veracidade de um experimento científico, assim como a sua real possibilidade de reprodução (BUNEMAN e KHANNA, 2001).

Um dos diferenciais do registro de proveniência do *MiningFlow* é a sua capacidade de armazenar os dados ontológicos em sua base de dados e relacioná-los com as definições dos *workflows* que por ventura sejam feitas. Desta maneira, podemos disponibilizar ao usuário dados de proveniência com semântica associada. Assim sendo, o usuário do ambiente estará apto a saber quais conceitos estão associados a um determinado *workflow* concreto ou a uma determinada execução de um *workflow*.

O módulo de proveniência possui três atividades principais que são executadas:

- **Armazenamento dos conceitos da ontologia**: esta fase se dá sempre que uma ontologia for registrada no sistema para uso. O módulo de registro de serviços invoca o módulo de proveniência para que o mesmo possa efetuar um reconhecimento da ontologia e mapear as classes, conceitos e relacionamentos para seu esquema relacional proposto.

- **Armazenamento da proveniência estática**: chamamos de proveniência estática as definições dos *workflows* (concreto e abstrato) que forem implementadas pelos usuários do sistema. Cada definição do *workflow* é armazenada em sua tabela associada, e ambas sempre são relacionadas aos conceitos da ontologia.

- **Armazenamento de proveniência dinâmica**: este tipo de proveniência se dá no momento da execução do *workflow* pelo SGWf. O módulo de proveniência tem como objetivo capturar os resultados intermediários e finais das atividades dos *workflows* concretos. Estes resultados podem ser importantes para futuras verificações e análises realizadas pelo módulo de consulta aos dados de proveniência.

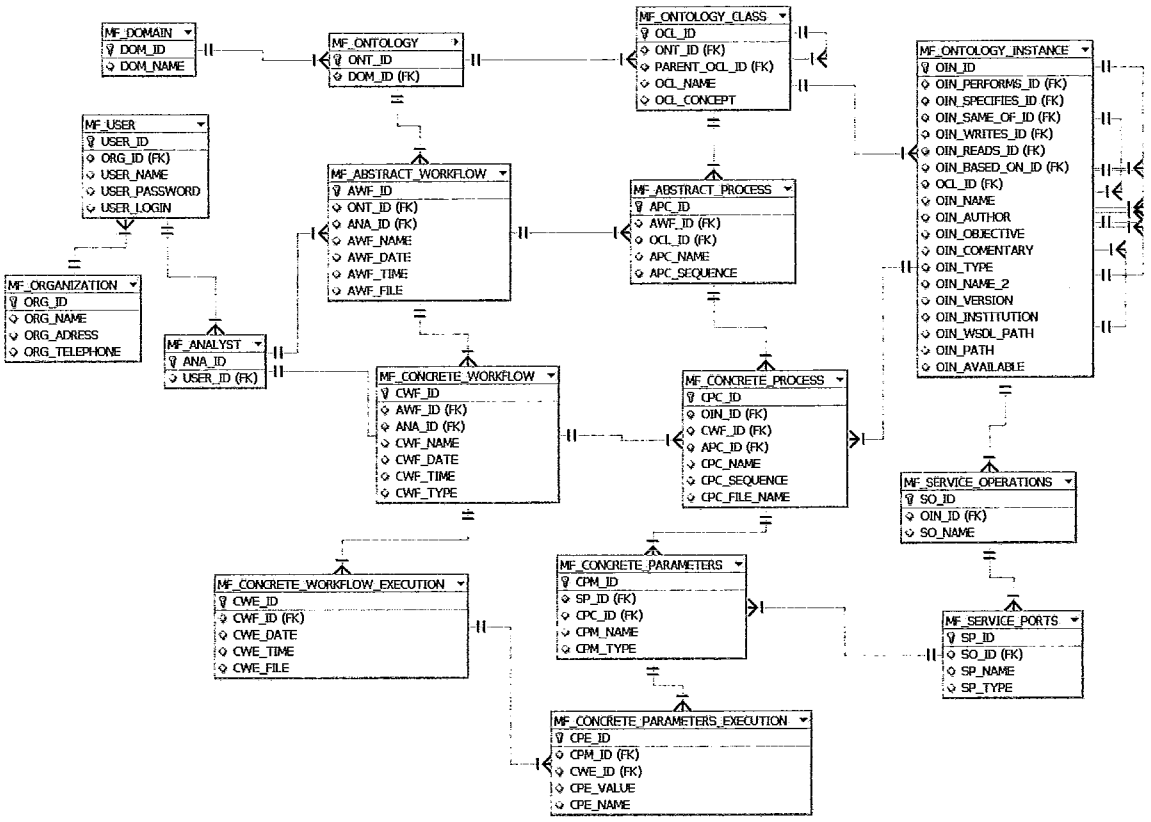


**Figura 4.5 Esquema de proveniência do *MiningFlow*.**

### 4.1.7 Módulo de consulta aos dados de proveniência

O módulo de consulta aos dados de proveniência tem como objetivo disponibilizar consultas pré-definidas pelo usuário para acesso aos dados de proveniência anteriormente armazenados. Este módulo deve ser invocado sempre que seja necessário recuperar informações da base de dados. Sua única função é disponibilizar os dados requeridos pelos usuários em um formato estruturado.

### 4.1.8 Limitações da Proposta

Uma das limitações conhecidas da proposta é a simplicidade dos módulos de definição existentes, no que se refere à estrutura permitida para os *workflows* definidos. Os módulos de definição (abstrata, concreta e de redefinição) não possuem controles de execução embutidos, como estruturas de repetição, laços de execução, estruturas de decisão, etc. O *MiningFlow* está preparado para trabalhar apenas com *workflows* DAG (YU e BUYYA, 2005), sem este tipo de estrutura embutida no mesmo. Estes controles já se fazem presentes em muitos dos SGWf atuais, mas visto que o processo de

mineração de textos possui um *workflow* "genérico" e relativamente simples (com as fases de pré-processamento, mineração e pós-processamento), esta limitação não se torna tão grave a ponto de invalidar a proposta.

Além disto, estas estruturas de controle não são encontradas em todos os SGWf existentes. Como o *MiningFlow* tem como objetivo oferecer um complemento aos SGWf existentes, e ao mesmo tempo ser independente de implementações existentes, partimos da premissa que a proposta deve gerar saídas para os mais diferentes SGWf. Existem formalismos para definir *workflows,* como os propostos por AALST, HOFSTEDE e BARROS (2000), e pode se tornar muito complicado definir controles no *MiningFlow* que possam ser traduzidos para os mais diferentes SGWf. Por exemplo, um laço de repetição pode ser definido no *MiningFlow*, porém não poderia ser traduzido para o VisTrails, já que o mesmo não disponibiliza esta estrutura em sua linguagem de definição. Assim sendo, optou-se por não disponibilizar estes controles na ferramenta, pois seria um complicador adicional no momento da tradução dos *workflows* abstratos e concretos para a linguagem da máquina de *workflow* escolhida. Conseqüentemente, o processo de tradução dos *workflows* se torna mais fácil e menos tendencioso a erros.

## 4.2 Participantes da Arquitetura

O ambiente *MiningFlow* descrito na subseção anterior possui basicamente quatro tipos de usuário que irão interagir nesta arquitetura. O **usuário publicador**, o **usuário que define os *workflows* abstratos** (os *templates* a serem seguidos para a definição dos *workflows* concretos), o **usuário que define os *workflows* concretos** (que executará o experimento na prática) e o **usuário que analisa os dados**.

Estas definições de usuários para o sistema são apenas papéis que cada pessoa pode assumir na arquitetura, entretanto, como em qualquer sistema de informação, as pessoas podem assumir privilégios sobrepostos na arquitetura.

O **usuário publicador** é aquele responsável por registrar na ontologia novos serviços ou programas e por registrar uma ontologia no sistema. Ele também é responsável por disponibilizar as aplicações científicas que serão usadas pelo ambiente.

O **usuário que define os *workflows* abstratos** normalmente será um analista de mineração de textos mais experiente. Esta pessoa deve possuir profundos

conhecimentos do ciclo de vida de um experimento de mineração de textos para que possa definir os *workflows* abstratos que possam servir como base para a definição dos experimentos executáveis (*workflows* concretos).

O **usuário que define o *workflow* concreto** é aquele que selecionará os serviços disponíveis na base do sistema para executar seu experimento. Ele deve seguir o *workflow* abstrato definido anteriormente. Este usuário é responsável por definir a seqüência de execução do experimento em termos de aplicações ou serviços que lhes são disponibilizados. Este usuário também definirá qual o nível de armazenamento dos dados de proveniência referente às execuções será necessário para cada experimento.

E finalmente o **usuário que analisa os dados**, cuja única função é verificar os dados que foram coletados em cada experimento de forma que se possa verificar se os experimentos foram válidos, por exemplo.

## 4.3 Ciclo de vida do experimento de mineração de texto

Uma vez descritos os componentes da arquitetura e os usuários que irão interagir com a mesma, temos que definir os passos de interação de cada usuário com a arquitetura no momento da definição de um experimento de mineração de textos no *MiningFlow*, ou seja, como um usuário pode definir o ciclo de vida de seu experimento utilizando o ambiente proposto.

O primeiro passo do ciclo de vida do experimento de mineração de textos dentro do *MiningFlow* acontece antes mesmo da definição do experimento propriamente dito. Este primeiro passo se refere à publicação de serviços pelo usuário publicador (Figura 4.6-A) e ao registro destes serviços na ontologia (Figura 4.6-B) e no ambiente *MiningFlow* (Figura 4.6-C).

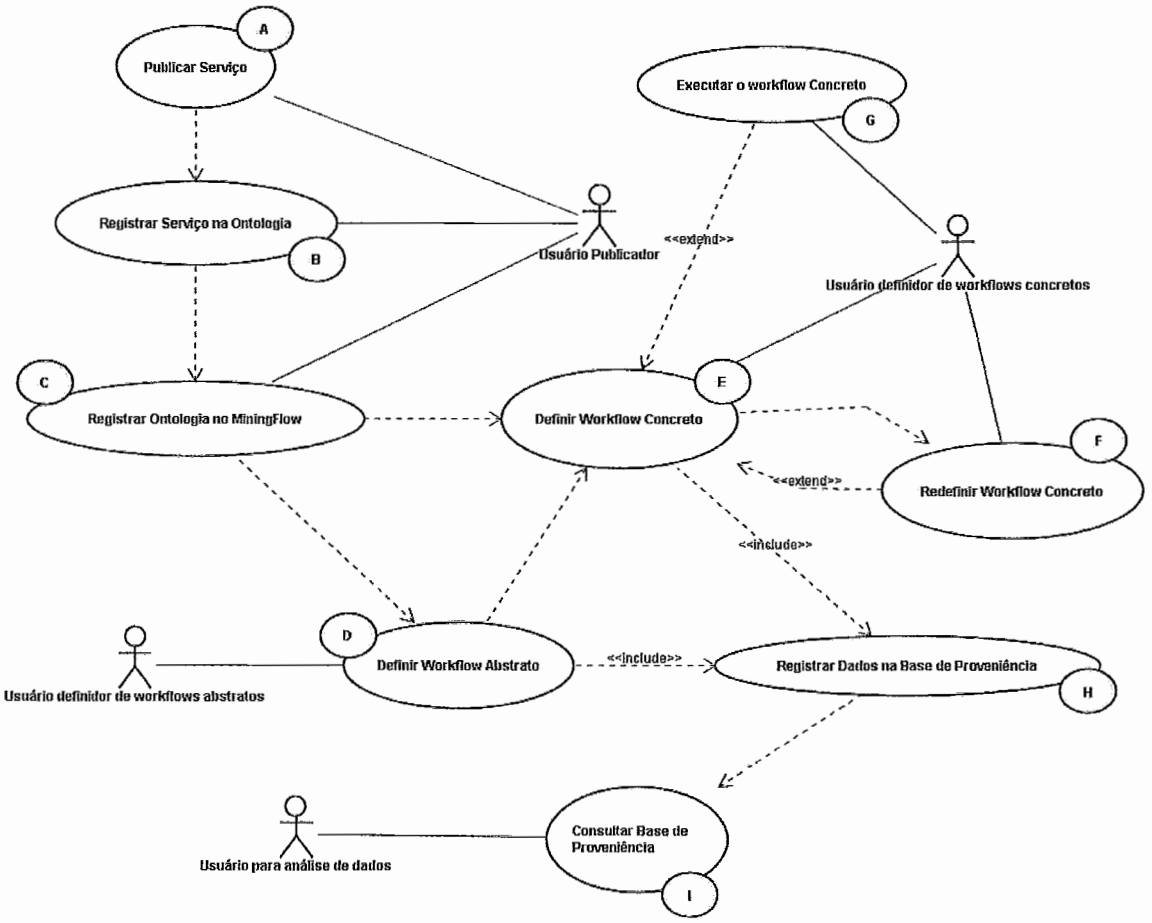


**Figura 4.6 - Diagrama de casos de uso para o ciclo de vida de um experimento de mineração de texto no *MiningFlow*.**

Após a fase de publicação do serviço, do registro do mesmo na ontologia e do registro da ontologia no ambiente, é chegada a hora de executar as etapas de definição do *workflow*. Como o *MiningFlow* propõe a utilização de duas definições para um mesmo *workflow* (abstrata e concreta), a atividade de definição do *workflow* foi desmembrada em duas atividades no diagrama de caso de uso. A primeira definição que é elaborada é a definição abstrata do experimento (Figura 4.6-D). Esta definição é realizada por um usuário experiente e associa para cada atividade do *workflow* uma classe da ontologia.

Uma vez definido o *workflow* abstrato, ficará disponível para os usuários menos experientes (usuários que definem os *workflows* concretos) a opção de definirem seus *workflows* concretos (Figura 4.6-E) a partir dos abstratos. Neste momento, cada usuário criará a sua definição do experimento em termos de serviços ou programas executáveis e não mais em termos de conceitos. Após esta definição, o usuário está apto a executar seu *workflow* concreto em um SGWf (Figura 4.6-G).

Após cada definição do *workflow* concreto, o usuário pode redefinir parâmetros de entrada sem alterar a estrutura do experimento, ou seja, quais serviços são utilizados e em que ordem (Figura 4.6-F). Esta redefinição é de extrema importância, pois por se tratar de um processo interativo e iterativo, o processo de mineração de textos tem a necessidade de executar diversas vezes um mesmo ciclo de mineração variando os parâmetros para que se possa avaliar a variação dos resultados obtidos.

Sempre que uma definição concreta ou abstrata for implementada (ou redefinida, no caso do *workflow* concreto), o módulo de registro de proveniência será invocado (Figura 4.6-H) para armazenar a definição do *workflow* em questão. Além disto, o módulo de proveniência sempre é invocado quando ocorre uma execução do *workflow* concreto pelo SGWf de forma que o módulo possa armazenar os dados intermediários obtidos na execução.

Uma vez executado, o *workflow* e seus dados associados estarão armazenados no banco de proveniência do ambiente. Desta forma, o usuário responsável pela análise dos dados poderá realizar consultas ao banco de proveniência (Figura 4.6-I) para validar experimentos ou recuperar execuções anteriores que lhe forem necessárias.

Todo o ciclo de vida do experimento, principalmente a fase de execução e de definição do *workflow* concreto, poderiam ser repetidas inúmeras vezes até que um resultado plausível (ou satisfatório) fosse alcançado.

Na realidade, todo o ciclo de vida do experimento é um processo interativo e incremental onde a cada execução pode ser necessário fazer alguns ajustes na própria definição do *workflow* ou até mesmo na criação de novas aplicações especificas para resolver o problema relativo ao experimento.

De qualquer maneira, o ambiente *MiningFlow* foi projetado de forma a ser o mais independente possível de qualquer SGWf ou de linguagem de definição de *workflows* e ao mesmo tempo flexível o suficiente para permitir alterações necessárias às novas realidades da área de pesquisa.

# Capítulo 5 A Implementação do *MiningFlow*

A proposta de arquitetura apresentada no capítulo 4 desta dissertação foi direcionada para manter as características de ser flexível e passível de adaptação do que se refere a cada módulo proposto. A arquitetura procura ser independente da tecnologia que será utilizada para implementar cada módulo componente da mesma. Entretanto, para o estudo de caso e o estudo de observação da proposta, algumas decisões importantes de implementação tiveram que ser tomadas. Estas decisões e suas vantagens e desvantagens são explicadas neste capítulo com detalhes.

A organização do texto do capítulo seguiu a distribuição dos módulos da arquitetura que foram previamente explicados no capítulo 4. Para cada módulo constante da arquitetura foi detalhada a sua implementação, e quais os problemas existentes e soluções adotadas no desenvolvimento do protótipo.

Além da explicação detalhada dos módulos componentes, será explicado o desenvolvimento de componentes auxiliares ao protótipo, de forma que os estudos de caso e de observação pudessem ser realizados de maneira satisfatória.

Uma das principais preocupações durante a implementação do *MiningFlow* foi a não utilização de linguagens ou ferramentas que fossem proprietárias. Assim sendo, optamos por utilizar sempre linguagens e ferramentas de código aberto.

Toda a implementação da arquitetura foi realizada em Java (SUN, 2007). O servidor de aplicação que é utilizado para disponibilizar os serviços *web* e o portal é o Apache Tomcat (TOMCAT, 2007). Ao Tomcat foi incorporado o pacote AXIS (2007) que tem como principal função transformar classes Java em serviços *web* e em atender as requisições SOAP.

Para interagir com a ontologia definida utilizamos a API disponibilizada pela ferramenta Protégé (PROTEGE, 2007) para trabalhar com arquivos OWL (W3C, 2007d).

Também se fez necessária a utilização de um sistema gerenciador de banco de dados (SGBD) para armazenar os dados de proveniência que foram colhidos durante a

definição e execução do experimento. O servidor MySQL (MYSQL, 2007) foi o escolhido para lidar com os dados de proveniência por ser o mais leve da categoria. Este SGBD tem limitações conhecidas pela comunidade, entretanto, para o experimento de observação e o estudo de caso ele se mostrou suficiente.

## 5.1 Desenvolvimento dos Módulos do *MiningFlow*

Esta seção tem como objetivo principal detalhar a implementação de cada módulo da arquitetura do *MiningFlow*. As subseções que se seguem espelham a divisão dos módulos mostrada no capítulo 4 desta dissertação .

### 5.1.1 Desenvolvimento do Módulo de Registro de Serviços

O módulo de registro de serviços é o único módulo que não se encontra acoplado ao portal *MiningFlow*. Para acessar este módulo, o usuário responsável por registrar os serviços e as ontologias (**usuário publicador**) deverá interagir com uma aplicação desenvolvida a parte.

Esta aplicação foi desenvolvida como uma classe Java simples que recebe um arquivo OWL com a ontologia definida como entrada, identifica os construtos desta ontologia (classes, relacionamentos, instâncias, restrições, etc.) e os carrega para a base de proveniência do *MiningFlow*.

Como na base de proveniência (apresentada no capítulo anterior) possuímos as entidades associadas aos construtos da ontologia, o módulo apenas se encarrega de identificar o construto e gerar um INSERT ou UPDATE na tabela associada do esquema relacional.

Apesar da proposta da arquitetura *MiningFlow* ser flexível a ponto de se usar qualquer ontologia acoplada à ferramenta, para a implementação do protótipo do estudo de observação, optamos por desenvolver o módulo de registro de serviços específico para a *MF-Ontology*. Desta forma, caso se use o módulo para carregar alguma outra ontologia para a base de dados, possivelmente ocorrerão erros durante a carga, comprometendo o processo como um todo.

O cadastramento de instâncias ou de novas classes na ontologia continua sendo realizado via Protégé (PROTEGE, 2007), sem que aja nenhuma ferramenta

desenvolvida para oferecer apoio a esta tarefa. O usuário deve primeiramente cadastrar seus serviços na ontologia e depois utilizar o módulo de registro de serviços para transferir as informações da ontologia para a base de proveniência.

Uma pergunta que pode ser feita no momento é o porquê de se cadastrar os conceitos da ontologia previamente na base de proveniência. Esta pergunta será respondida nas próximas subseções.

### 5.1.2 Desenvolvimento do Módulo de Definição Abstrata do *Workflow*

O módulo de definição abstrata do *workflow* foi desenvolvido como um *servlet* Java acoplado ao portal do *MiningFlow* e possui duas operações principais que são invocadas pelo portal: a configuração do *workflow* abstrato e a definição do *workflow* abstrato. A primeira operação configura alguns parâmetros necessários para que se defina o *workflow* abstrato. Nesta primeira etapa, o usuário escolherá um nome para o *workflow*, a quantidade de atividades do *workflow* e qual ontologia estará associada ao *workflow* abstrato que está sendo gerado.

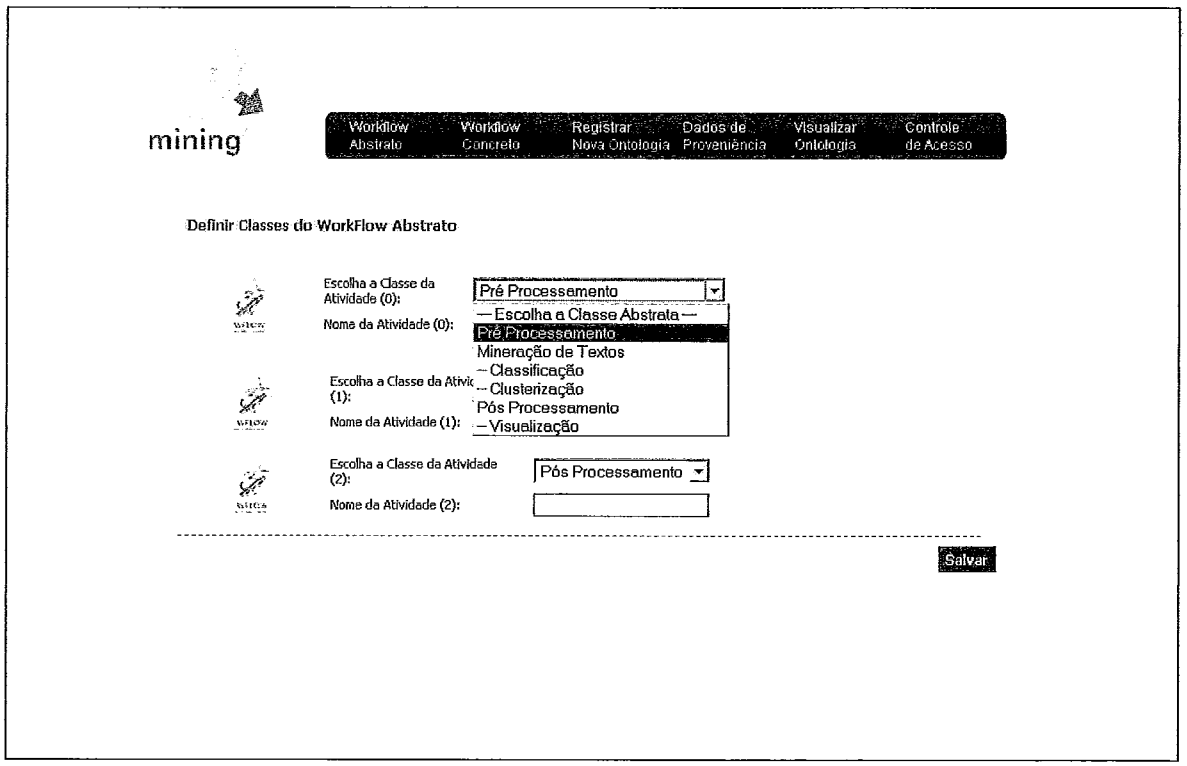


**Figura 5.1 Operação de configuração do *workflow* abstrato.**

A figura 5.1 nos mostra a segunda operação disponibilizada pelo módulo que é a definição do *workflow* abstrato. Nesta etapa, as classes definidas na ontologia e

previamente carregadas na base de dados de proveniência, são exibidas para o usuário, a fim de que o mesmo associe os conceitos exibidos com as atividades que compõem o *workflow*.

Optamos por carregar os conceitos da ontologia da base de dados ao invés de acessar diretamente o arquivo OWL via *plugin,* pois o acesso ao arquivo pelo *plugin* do Protégé se mostrou falho em alguns momentos. Após pesquisas, descobrimos que falhas intermitentes de acesso são geradas quando se utiliza o *plugin* do Protégé através de uma pagina *web*. Principalmente por esta razão resolvemos mapear os conceitos da ontologia para um esquema relacional e utilizar somente os dados carregados nesta base de proveniência. A solução adotada não chega a ser um problema, pois de qualquer maneira teríamos que carregar os conceitos da ontologia na base de dados para que pudéssemos associar a definição do *workflow* abstrato com as classes ontológicas. O que fizemos, portanto, foi adiantar uma atividade que já se mostrava inevitável durante o processo.

### 5.1.3 Desenvolvimento do Módulo de Definição Concreta do *Workflow*

O módulo de definição concreta do *workflow*, assim como o módulo de definição abstrata foi desenvolvido como um *servlet* Java acoplado ao portal do *MiningFlow* e possui quatro operações principais que são invocadas pelo portal: a seleção do *workflow* abstrato que servirá de base, a seleção dos serviços associados a cada atividade, a escolha das operações dos serviços e a definição dos parâmetros de entrada de cada serviço *web*.

O primeiro passo para o usuário é a seleção do *workflow* abstrato que servirá de base para a confecção do *workflow* concreto. Uma vez selecionado um *workflow* abstrato já definido, os conceitos associados a cada atividade do *workflow* serão recuperados de forma que sejam disponibilizados serviços para serem escolhidos para cada tarefa (Figura 5.2). A definição abstrata é recuperada da base de proveniência do *MiningFlow*.

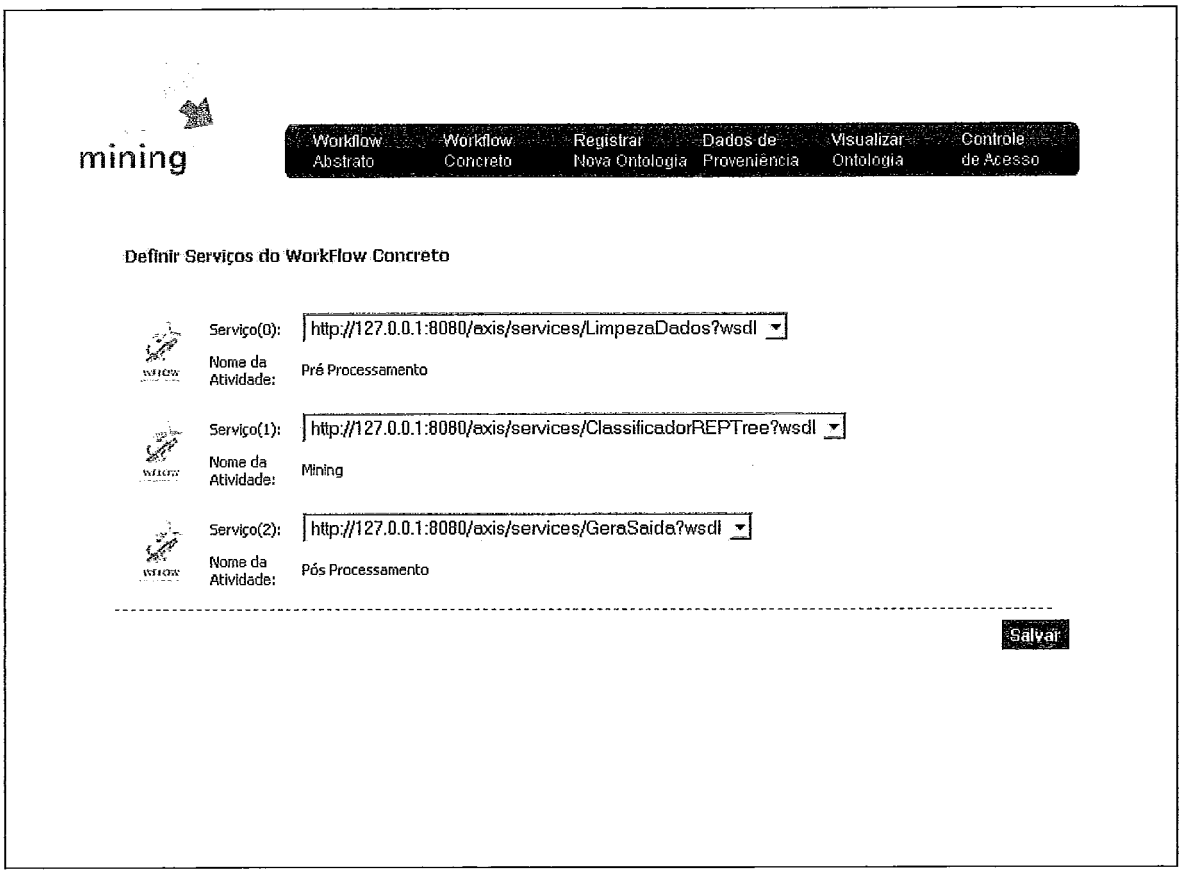


**Figura 5.2 Operação de configuração do *workflow* concreto (seleção dos serviços).**

Após selecionarmos cada arquivo WSDL que estará associado a cada atividade do *workflow* (Figura 5.2), temos que escolher qual operação do serviço *web* que iremos utilizar. É importante ressaltar que um mesmo serviço *web* pode disponibilizar uma série de operações diferentes e por isso, o usuário deve escolher a que melhor lhe convir.

Após selecionar qual operação do serviço *web* será executada, o módulo pode executar a última operação disponível. Os parâmetros de entrada devem ser configurados de forma que o *workflow* concreto possa ser executado por algum SGWf.

Cada parâmetro de cada operação selecionada é exibido na tela, para que o usuário esteja apto a informar valores de entrada. Todos os parâmetros de saída dos serviços não são editáveis pelo usuário, uma vez que estes parâmetros são gerados pelo serviço e servem apenas como entrada para o próximo serviço na seqüência de execução do *workflow*.

Os parâmetros de entrada são informados pelo usuário, exceto quando um serviço possuir apenas um parâmetro de entrada. Quando o parâmetro de entrada for único, o mesmo fica desabilitado para edição, pois obrigatoriamente ele estará conectado ao parâmetro de saída do serviço anterior do *pipeline*.

### 5.1.4 Desenvolvimento do Módulo de Redefinição do *Workflow*

O módulo de redefinição do *workflow* concreto é o responsável por possibilitar ao usuário alterar parâmetros de entrada sem modificar a forma do *workflow*, ou seja, quais serviços que irão compor o *pipeline*.

Este módulo simplesmente disponibiliza duas operações: a seleção de um *workflow* concreto já definido e a configuração dos parâmetros. Após selecionar em uma listagem um *workflow* concreto já definido, cada parâmetro de cada operação do *workflow* concreto escolhido é exibido na tela, para que o usuário esteja apto a informar valores de entrada. Todos os parâmetros de saída dos serviços não são editáveis pelo usuário, uma vez que estes parâmetros são gerados pelo serviço e servem apenas como entrada para o próximo serviço na seqüência de execução do *workflow*.

### 5.1.5 Desenvolvimento do Módulo de Tradução

O módulo de tradução de *workflows* concretos é uma das grandes vantagens da *MiningFlow*. Este módulo tem como objetivo único e fundamental de receber uma definição concreta (ou redefinição) em termos de classes do sistema e convertê-la para uma saída compreensível por algum SGWf. Por motivos de simplificação da implementação, este módulo executa a tradução da definição do *workflow* em duas fases distintas:

- A **primeira fase** se propõe a identificar todos os componentes dos *workflow* que não sejam as ligações entre atividades (ou seja, serviços *web*, leitores de arquivo, componentes geradores de relatórios, etc.). Uma vez identificados os componentes, o módulo verifica qual a opção de SGWf que foi informada e faz a tradução de cada componente para as *tags* XML de cada linguagem que possui um tradutor implementado.

- A **segunda fase** tem como objetivo identificar as ligações entre os componentes do *workflow*, mapeando, por exemplo, qual a porta de saída

de um serviço que se ligará na porta de entrada do próximo. Esta fase poderia ser mais custosa caso as informações dos serviços (portas, operações e tipos de dados) já não tivessem sido pré-armazenados no banco de proveniência. O módulo teria que acessar o WSDL de cada serviço para poder identificar as portas e operações, o que (dependendo da conexão) pode ser muito mais lento do que um acesso a um registro no SGBD.

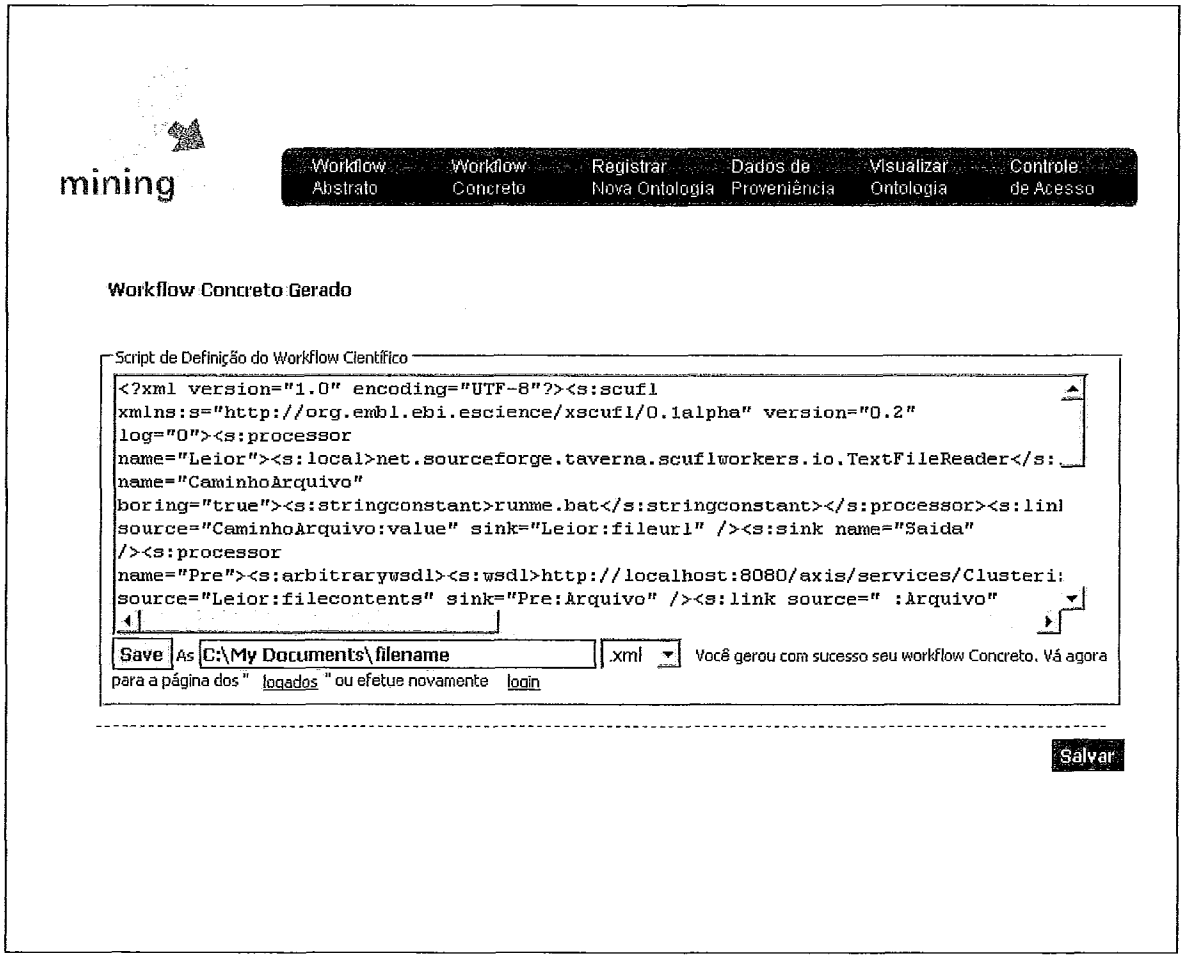


**Figura 5.3 Resultado da definição do *workflow* concreto.**

Ao final do processo de tradução, o módulo executa duas tarefas: inclui os serviços de proveniência dinâmica em cada atividade (esta operação será melhor detalhada na subseção 5.1.6) e exibe na interface o XML de definição do *workflow* concreto já traduzido para interpretação do SGBD (Figura 5.3). Uma evolução deste módulo é a criação de uma chamada para execução pelo SGWf automaticamente caso o usuário já possua o SGWf instalado na sua máquina.

### 5.1.6 Desenvolvimento do Módulo de Registro de Proveniência

O módulo de registro de proveniência tem como finalidade principal capturar os dois tipos de proveniência já explicados no capitulo 4: a proveniência estática (parâmetros e definições abstratas e concretas) e a proveniência dinâmica (resultados intermediários e finais).

Justamente por este motivo, este módulo foi dividido em dois sub-módulos diferentes. O primeiro sub-módulo tem como objetivo capturar as definições abstratas e concretas e registrá-las na base de proveniência. Para tal foi implementada uma classe Java que recebe como parâmetro de entrada uma coleção de objetos que definem o *workflow* abstrato e concreto na implementação interna do sistema. Estes objetos são mapeados para as entidades do nosso esquema relacional de forma que as definições sejam persistidas no banco de dados. Este registro acontece sempre que um *workflow* concreto ou abstrato é definido (ou redefinido).

O segundo sub-módulo de proveniência é o responsável por registrar a proveniência dinâmica. Este sub-módulo fica localizado em um servidor e é disponibilizado através de uma série de serviços *web* diferentes. Este tipo de serviço facilita a integração entre os diversos usuários do *MiningFlow*. Cada atividade (serviço) que compõe o *workflow* monta seu próprio pacote SOAP e envia as informações através de uma conexão HTTP comum para o serviço de proveniência.

Estes serviços foram implementados em Java com Apache Axis de acordo com a especificação de serviços *web* do W3C. Para cada pacote SOAP que chega ao serviço é criada uma *thread* para salvar as informações no banco de dados, evitando-se assim a perda de dados importantes.

Foram implementados três serviços inicialmente para proveniência. Como os serviços *web* que utilizamos e implementamos retornavam apenas três tipos de dados (inteiros, string e arquivos XML) foram criados serviços que recebiam apenas estes tipos de dados como entrada.

O módulo de tradução incorpora na saída de cada serviço do *pipeline* um novo serviço de proveniência que registra no esquema relacional o resultado da execução do serviço. Conseqüentemente, o *workflow* final gerado irá possuir uma série de serviços

de proveniência acoplados ao mesmo, e que serão visíveis para o usuário. Desta forma, o *workflow* final poderá ser considerado por alguns como "poluído", uma vez que possuirá serviços que não foram previamente definidos pelos usuários.

O módulo de tradução verifica qual o tipo de saída do serviço *web* (inteiro, string, XML) e incorpora o serviço apropriado para cada situação. No momento da execução do *workflow* pelo SGWf, o serviço *web* de proveniência registrará a saída da atividade executada no banco relacional em um campo *BLOB* que pode ser recuperado *a posteriori* pelos usuários.

### 5.1.7 Desenvolvimento do Módulo de Consulta de Proveniência

O módulo de consulta de proveniência foi, talvez, o mais simplificado durante a fase de implementação, uma vez que esta é uma área de pesquisa ativa, que caminha na direção da definição de padrões tanto para a representação quanto para a consulta de dados de proveniência. Assim, o objetivo do módulo junto a *MiningFlow* é o de prova de conceito e não o de contribuição na definição de esquemas de proveniência nem tampouco modelos de captura e recuperação de dados de proveniência. Este módulo se resume em dois principais componentes principais, descritos a seguir.

Uma classe que acessa diversas visões pré-definidas no banco de dados e exibe o resultado de uma consulta SQL (também pré-definida) na interface *web*. Por exemplo, uma das visões criadas exibe quantas execuções de um determinado *workflow* concreto foram realizadas. Por acessar diretamente um objeto do SGBD, a alteração deste módulo se torna simples, bastando inserir um comando SQL dentro da classe que implementa o módulo por completo.

Um *Servlet* que possibilita ao usuário recuperar resultados intermediários já armazenados. Este *Servlet* necessita que sejam passados para ele dois parâmetros: um *workflow* concreto e uma data e hora de execução do mesmo (visto que um mesmo *workflow* concreto pode possuir *n* execuções diferentes).

Após selecionar um *workflow* e uma execução do mesmo, o usuário receberá como resposta uma tabela com as atividades do *workflow* escolhido e os resultados intermediários que foram colhidos na execução selecionada. Na interface o usuário deverá selecionar quais dados deseja recuperar e informar um prefixo para os arquivos,

uma vez que os arquivos são salvos no SGBD em formato binário e não possuem mais nomes ou extensões. Uma vez selecionados os arquivos e configurado o prefixo, o usuário poderá baixar os arquivos para a sua máquina (já que os mesmos são criados em uma pasta no servidor de aplicação).

## 5.2 Desenvolvimento do Portal *MiningFlow*

Para possibilitar a criação e o acompanhamento dos experimentos de mineração de texto (e a conseqüente invocação dos módulos da arquitetura) foi desenvolvido um portal *web* (disponível em www.cos.ufrj.br/~danielc) para disponibilizar o meio para o usuário definir, redefinir e executar o seu *workflow* construído. Dos quatro tipos de usuários descritos no capítulo 4, três utilizaram o portal como forma de trabalho.

O portal *MiningFlow* é composto por uma série de páginas HTML (W3C, 1999) e *servlets* que propiciam uma interface visual (Figura 5.4) simples para apoiar o ciclo de vida de um experimento de mineração de textos. Alguns exemplos das atividades que o portal apóia: definição do *workflow* abstrato, definição do *workflow* concreto, redefinição de um *workflow* concreto, visualização dos dados de proveniência, entre outros.

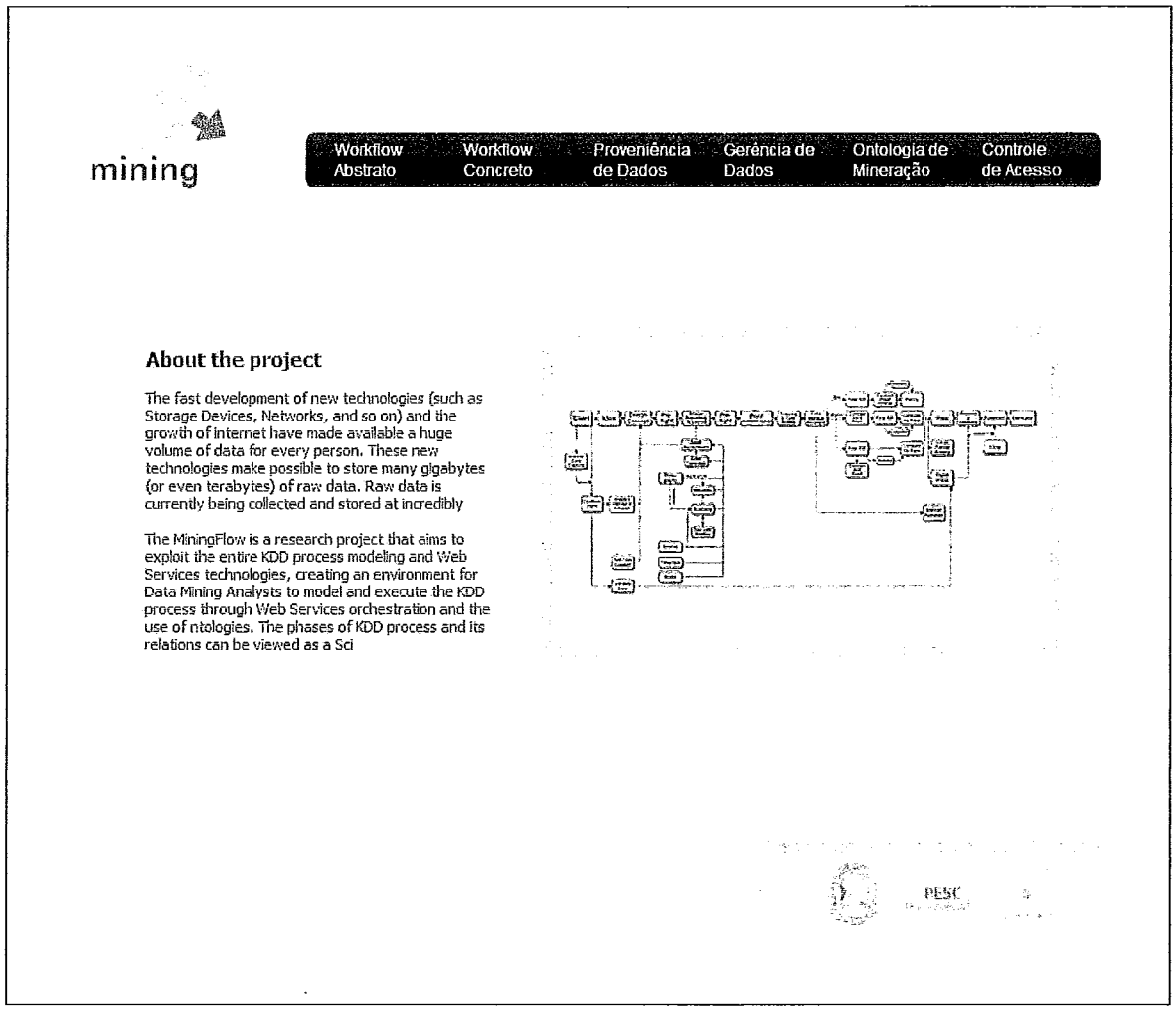


**Figura 5.4 Tela principal do portal *MiningFlow*.**

A grande vantagem de se utilizar um portal de desenvolvimento de experimentos é que os componentes da arquitetura se encontram disponíveis para uma gama maior de pessoas, além de se encontrarem centralizados e com acesso público, sem a necessidade de configurar ou alterar alguma configuração já existente na máquina do usuário.

Além disto, utilizando-se um portal e serviços *web* acoplados, podemos acessar as aplicações pelo navegador de sítios *web* que quisermos, seja ele o Internet Explorer (MICROSOFT, 2007), ou o Mozilla Firefox (MOZILLA, 2007), por exemplo. Além disto, através do portal, ganhamos em portabilidade já que poderemos acessá-lo de máquinas com Microsoft Windows (MICROSOFT, 2007), Mac OS X (APPLE, 2007) ou Linux (LINUX, 2007).

## 5.3 Serviços *Web* de Aplicações Científicas

No domínio em que estamos trabalhando (mineração de textos), inicialmente, nenhum dos programas que são utilizados pelos analistas estavam disponíveis como serviços *web*.

Um primeiro desafio, portanto, foi transformar programas usados pelos analistas de mineração de texto (sendo que muitos dos programas não possuem nem API definida) em serviços *web* que pudessem ser acessados pela *Intranet* ou *Internet* e pudessem ser incorporados à ferramenta.

Para simplificar a implementação, foram desenvolvidos alguns adaptadores (*wrappers*) que serviram como mediadores entre aplicações que já existiam (aplicações legadas) e a interface de serviço para os usuários externos. Muitos dos serviços *web* que foram desenvolvidos tinham como função apenas invocar a aplicação legada passando alguns parâmetros de entrada e retornar o resultado obtido com a execução. Um exemplo de uma classe Java que gerou um serviço *web* adaptador pode ser encontrada na figura 5.5.

```
public class ClassificadorREPTree {

    /**
     * @param args
     */
    public static String REPTreeGraph(String Arquivo, String Atributos){
        String tempFileName = "REPTree_"+Math.abs((new Random()).nextLong())+".arff";
        String caminhoArquivo = "C:\\"+tempFileName;

        try {
                BufferedWriter out = new BufferedWriter(new FileWriter(caminhoArquivo));
                out.write(Arquivo);
                out.close();

            } catch (IOException e) {
            }

          DataHandler dh = new DataHandler(new FileDataSource(caminhoArquivo));
          REPTree rp = new REPTree();
          String result =  rp.classifyGraph(dh,Atributos);
        return result;
    }

}
```

**Figura 5.5 Exemplo de classe de implementação de serviço *web*.**

A classe exibida na figura 5.5 tem como finalidade receber um arquivo de entrada e uma *string* com os atributos a serem considerados e retornar uma *string* contendo uma definição de um grafo que implementa a visualização de uma árvore de decisão. Esta implementação de uma árvore de decisão foi obtida da ferramenta Weka (WITTEN e FRANK, 2005), assim como outras implementações que foram disponibilizadas como serviços *web* no ambiente *MiningFlow*.

Após a implementação da classe adaptadora, a mesma foi submetida à biblioteca Axis para que os serviços *web* fossem gerados e ficassem disponíveis para a invocação pelo portal *MiningFlow*. O Axis gera um arquivo WSDL (W3CNOTE, 2007) que disponibiliza uma interface pública para acesso ao serviço.

Cada serviço *web* que foi implementado (sendo baseado em uma aplicação legada ou não) foi incorporado ao ambiente através do módulo de registro de serviços, detalhado na subseção 5.1.1 deste capítulo.

## 5.4 Desenvolvimento das Funcionalidades de Apoio

Algumas funcionalidades foram implementadas para oferecer apoio ao usuário nas tarefas de definição e execução dos seus *workflows*.

Como os experimentos normalmente envolverão uma coleção de documentos de tamanho considerável, pode se tornar inviável enviar e receber essas coleções pela rede. Assim sendo, o processamento se torna mais eficiente quando os arquivos a serem processados já se encontram na mesma máquina do serviço *web*, ou seja, o servidor de aplicação.

Para tal, foram desenvolvidas interfaces que possibilitam realizar a carga, a recuperação e o gerenciamento destas coleções em diretórios do servidor de aplicação

# Capítulo 6 Avaliação do *Miningflow*

Nos capítulos 4 e 5 desta dissertação foram apresentadas a arquitetura proposta para o *Miningflow* e a implementação da mesma, onde cada um dos seu módulos, e a relação entre os mesmos, foram detalhadamente explicados. Para avaliar a viabilidade da proposta, foi implementado um protótipo funcional e elaborado um estudo de caso, juntamente com um estudo de observação para avaliação da ferramenta e da proposta em si.

Neste capítulo será descrita toda a elaboração do estudo de caso de avaliação do *Miningflow* (e do estudo de observação), bem como as conclusões alcançadas com a utilização do mesmo.

Segundo Travassos (2005), a realização de um estudo experimental de avaliação pode ser dividida em cinco fases principais: a **definição**, o **planejamento**, a **execução**, a **análise** e o **empacotamento** do estudo.

A etapa de definição tem como objetivo, fazer com que o avaliador defina claramente os objetivos do estudo e quais os objetos que serão analisados pelo mesmo. O planejamento consiste em descrever o perfil dos participantes, dos instrumentos necessários para a execução do experimento e o processo de execução em si. A etapa de execução, como o próprio nome já nos diz, consiste na realização do estudo experimental com os participantes, utilizando o ferramental que foi previamente definido na fase de planejamento. A etapa de análise consiste na avaliação dos resultados obtidos na etapa de execução e a conseqüente organização destes resultados para melhor compreensão. A etapa final, o empacotamento, consiste em armazenar os dados e documentos obtidos, de forma a repetir mais facilmente o experimento no futuro.

As etapas acima descritas serão explicadas conforme as caracterizarmos para o processo de avaliação do *Miningflow*. Os tópicos que se seguem neste capítulo detalham cada etapa do processo de avaliação, contextualizando-o para o estudo de caso da proposta. Os gráficos relacionados às avaliações dos usuários (tanto do estudo de observação quanto do estudo de caso) se encontram no Anexo D desta dissertação.

## 6.1 Instanciação do *Miningflow*

Antes da execução efetiva do estudo de observação e do estudo de caso, algumas instanciações foram realizadas na ferramenta para que os estudos fossem possíveis.

Nos estudos iniciais e no projeto piloto do sistema foram desenvolvidos diversos serviços *web* baseados no WEKA. Entre eles um serviço que lia um arquivo ARFF (padrão WEKA), um serviço com a implementação do algoritmo de agrupamento *EM*, um classificador *J4.8*, um clusterizador *simple K-Means* e um serviço de proveniência.

Apesar destas implementações não estarem voltadas para o domínio de mineração de textos, este estudo inicial visava analisar a ferramenta e suas funcionalidades em um estado muito incipiente, logo, como estes serviços já haviam sido criados anteriormente durante o período de elaboração da dissertação, optou-se por usá-los em uma primeira avaliação. Um detalhamento maior da implementação destes serviços e do funcionamento de algoritmos pode ser encontrado em WITTEN e FRANK (2005).

Cada implementação destes serviços foi devidamente cadastrada na ontologia e estes termos foram carregados para o *Miningflow* através do módulo de carga.

Para os estudos de observação e de caso do *Miningflow,* os serviços adaptados do WEKA continuaram cadastrados na base e novos serviços foram implementados. Estes novos serviços *web* implementados foram baseados em programas enviados por alunos do PEC. Os novos serviços *web* gerados foram: um serviço de remoção de *stop words*, um serviço de *stemmização*, um serviço de geração de *Bag Of Words* (BOW) (gerando uma saída tanto em padrão "separado por vírgulas" quanto em padrão ARFF), um serviço de limpeza da coleção (este serviço capta uma coleção inteira de documentos e retira espaços desnecessários ou caracteres desconhecidos) e um serviço de clusterização *K-Means*.

Além destes serviços, foram criados novos serviços, que eram variações dos serviços iniciais. Por exemplo, um serviço *web* que englobava todo a etapa de pré-processamento (e conseqüentemente englobava os serviços já citados anteriormente). Todos estes serviços foram cadastrados na ontologia (*MF-Ontology)* e carregados para o

*Miningflow*. Estes passos de registro e carga foram executados pelo experimentador, sem que o usuário necessitasse intervir.

Assim sendo, para os estudos de observação e de caso, todas as classes da ontologia estavam disponíveis no momento da definição do *workflow* abstrato, porém, no momento da definição do *workflow* concreto, somente as classes que possuíam serviços associados exibiram resultados e habilitaram o usuário a definir seu *workflow* concreto.

## 6.2 Estudo de Observação

Seguindo a proposta da definição de um processo de avaliação definida por Pereira (2007), um primeiro estudo de observação foi proposto e realizado em um ambiente controlado, com apenas um servidor e participantes com perfis aleatórios (participantes especialistas e não-especialistas no domínio de mineração de textos). Este experimento foi supervisionado pessoalmente pelo experimentador. Através deste experimento, conseguiu-se extrair informações quanto à execução da ferramenta a fim de corrigir algumas falhas de programação que ainda existiam, além de executar uma primeira avaliação da proposta.

### 6.2.1 Definição do Estudo de Observação

O objetivo deste estudo de observação foi avaliar a viabilidade de utilização do *Miningflow*, ou seja, avaliar o grau de usabilidade e as dificuldades das funcionalidades do *Miningflow* para a criação de uma aplicação de mineração de textos através da utilização de tecnologias de *workflow* e ontologias. Os usuários avaliaram a ferramenta através de questionários, indicando pontos fortes, fracos e possíveis melhorias da mesma. Como esta foi uma avaliação preliminar, muitas das melhorias propostas foram posteriormente implementadas para o estudo de caso.

Portanto, um protótipo do *Miningflow* foi implantado na tentativa de validar, principalmente, a hipótese de que podemos criar uma camada intermediária (*front-end*) que apóie o usuário final a definir suas aplicações de mineração de textos utilizando a ontologia de domínio proposta juntamente com o modelo de separação em *workflows* abstratos e concretos baseados na proposta de Cavalcanti (2003).

Outro ponto importante que devemos destacar é a avaliação das dificuldades encontradas pelos usuários durante o processo de definição dos *workflows* abstratos, concretos e redefinições com a utilização da ferramenta.

Desta forma, este estudo de observação proposto foi desenvolvido e aplicado a uma série de usuários que serão descritos a seguir. É importante ressaltar que este estudo, não está interessado em avaliar ou calcular o ganho de tempo derivado da utilização do *Miningflow* em relação ao modo atual como os experimentos são desenvolvidos, mas sim, considerar a viabilidade de utilização e que melhorias podem ser definidas para o processo de definição de um *workflow* para mineração de textos.

Utilizando a notação baseada em GQM (VAN SOLINGEN, BERGHOUT, 1999), temos que:

"**Analisar** a utilização do sistema *Miningflow*; **com o propósito de** avaliar a viabilidade de sua utilização na geração de *workflows* (concreto e abstrato) **aplicados no** domínio de mineração de texto; **referente** à satisfação do usuário **do ponto de vista do** aprendiz e do especialista".

#### 6.2.1.1 Perfil dos Participantes

Apesar da proposta do *Miningflow* ter sido implementada através de um portal na *web*, os participantes do experimento podem (ou não) estar localizados em locais diferentes (todos os participantes podem estar em uma única sala, por exemplo). Entretanto, para este primeiro estudo de observação é interessante que se faça o acesso via *web* com participantes localizados no mesmo local, porém sem contato uns com os outros de forma que o experimento não sofra interferências. Assim sendo, cada um dos participantes deve possuir uma conexão *web* estável para acessar o servidor da aplicação e do banco de dados. Deste modo, é necessário que os participantes tenham conhecimentos básicos sobre utilização do computador e uma mínima experiência em acesso a *web*. Como os participantes do experimento são da área de ciência da computação, este requisito foi facilmente atendido por todos.

É necessário também que os participantes tenham algum conhecimento sobre o domínio da aplicação, Caso contrário, pode ser necessário que alguns aspectos do processo sejam esclarecidos (é proposta uma "aula" de nivelamento para que os

usuários fiquem todos no mesmo patamar de conhecimento) para que esses participantes usem de forma apropriada às funcionalidades do sistema.

O estudo de observação contou com a participação de cinco voluntários, dentre eles, profissionais da área de computação (graduandos e mestres), e alunos de mestrado do PESC da COPPE/UFRJ.

De acordo com Tichy (2007), utilizar estudantes como participantes de estudos de avaliação experimental é aceitável caso eles sejam apropriadamente treinados e os dados usados para o experimento sejam direcionados para que possam realizar tarefas específicas e bem determinadas pelo experimentador. Estas condições foram seguidas nesta dissertação.

#### 6.2.1.2 Recursos Utilizados

Para que o experimento pudesse ser realizado, foi disponibilizado um computador Pentium Core II Duo 3.4 Ghz com 2 Gb de Memória RAM e HD de 500 Gb conectado a *internet* através de uma conexão a cabo de 2Mbit e funcionando como o servidor da aplicação e de banco de dados para o *Miningflow*.

Além do acesso à ferramenta, foram disponibilizados dois questionários impressos para os participantes do experimento:

1. Um questionário para qualificação dos participantes constante do Anexo A desta dissertação;

2. Um questionário para avaliação qualitativa do *Miningflow* constante do anexo B desta dissertação.

Nesta abordagem que estamos dispostos a utilizar, propomos que o *Miningflow* seja avaliado qualitativamente pelos usuários de acordo com as perguntas disponibilizadas em um questionário previamente elaborado.

A experiência tanto em relação à mineração de texto quanto em relação às tecnologias de *workflows* serão avaliadas pelo questionário antes do início do experimento de observação (o questionário se encontra no Anexo A desta dissertação). Após o término do experimento, é solicitado que os participantes respondam o

questionário elaborado para avaliação da ferramenta (questionário encontra-se no anexo B desta dissertação). As questões foram definidas considerando as características específicas do processo de mineração de textos em conjunto com as técnicas de *workflows* e o uso de ontologias.

Nas seções que se seguem, é apresentada a descrição do estudo de observação elaborado. A descrição está dividida em três partes principais a serem avaliadas: a primeira visa à definição de um *workflow* abstrato (*template*) que será utilizado para guiar usuários menos experientes. A segunda parte visa à definição do *workflow* concreto e a terceira à redefinição de um *workflow* concreto, habilitando a mudança de parâmetros do *workflow*.

#### 6.2.1.3 Projeto Piloto

Um projeto piloto anterior ao estudo de observação foi realizado com apenas dois avaliadores a fim de verificar a navegação no sistema e conseguir capturar eventuais problemas que pudessem ocorrer e que atrapalhassem a execução do mesmo. Este projeto piloto não sofreu avaliação formal por parte dos experimentadores.

#### 6.2.1.4 Treinamento dos Usuários

O treinamento dos usuários para o estudo de observação foi realizado presencialmente e ministrado pelo avaliador. Uma "aula de nivelamento" foi dada, e os conceitos principais de mineração de textos e *workflows* científicos foram abordados.

Numa visão geral, o treinamento foi composto de três rodadas (que seguem as fases do planejamento do experimento): geração do *workflow* abstrato, geração do *workflow* concreto e redefinição do *workflow*.

A capacidade de generalização deste estudo é discutida nas seções que se seguem, onde realizamos um estudo sobre os problemas que podemos encontrar ao avaliar uma ferramenta desta maneira.

### 6.2.2 Planejamento do experimento

O experimento foi composto de três partes principais: definição do *workflow* abstrato, definição do *workflow* concreto e redefinição de um *workflow* concreto. Cada

participante recebeu informações genéricas sobre as tarefas a serem executadas de forma que pudessem avaliar o quanto o sistema ajudou na geração de um *workflow*. A seguir é detalhada cada parte do experimento.

#### 6.2.2.1 Parte 1: Definição do *Workflow* Abstrato

Cada participante gerará um *workflow* abstrato que será usado como base para a geração dos *workflows* concretos a seguir. O participante terá que definir a quantidade de tarefas que irão compor o *workflow* e quais as classes da ontologia que estarão associadas a cada tarefa, escolhendo classes mais abrangentes (que englobem etapas completas do processo de mineração de textos) ou classes mais especializadas (que se restrinjam a sub-etapas, também chamadas de tarefas e/ou funções do processo). Realizando esta associação, nós definimos em uma visão macro, qual conceito determinada tarefa está representando.

#### 6.2.2.2 Parte 2: Definição do *Workflow* Concreto

Cada participante utilizará um *workflow* abstrato gerado por um companheiro para gerar um *workflow* concreto. A idéia é fazer com que o participante (2) utilize o *workflow* abstrato do participante (1), o participante (3) utilize o do participante (2) e o do participante (n) utilize o *workflow* abstrato definido pelo participante (1). Desta forma, cada participante utilizará um *template* definido por um companheiro sem conhecer as características do *workflow* abstrato que foi gerado. Desta maneira, podemos validar se a ferramenta realmente consegue "guiar" o usuário nesta definição.

#### 6.2.2.3 Parte 3: Redefinição do *Workflow* Concreto

Cada participante recuperará um *workflow* concreto gerado e redefinirá seus parâmetros, gerando uma nova versão do *workflow*. É importante ressaltar que nesta fase, o participante apenas redefinirá parâmetros, sem ter que escolher serviços e/ou operações a serem executadas. Em experimentos de mineração de textos, esta fase é de extrema importância, uma vez que um processo de mineração de textos pode ter a necessidade de ser executado inúmeras vezes (variando os parâmetros de entrada) até que se atinja um resultado satisfatório.

#### 6.2.2.4 Critérios de Avaliação

Para avaliar a proposta *Miningflow* se faz necessário que avaliemos os ganhos obtidos pelos usuários do sistema e também as dificuldades encontradas pelos mesmos. Para que pudéssemos realizar esta avaliação, utilizamos o processo de avaliação qualitativa. A satisfação dos participantes foi avaliada de acordo com um questionário que foi distribuído para os mesmos.

#### 6.2.2.5 Capacidade Aleatória

A capacidade aleatória é definida como o processo de escolha aleatória dos indivíduos que executarão o experimento dentre um universo maior de indivíduos que tem capacidade para executar o experimento. A capacidade de um determinado indivíduo executar o experimento pode ser avaliada segundo um questionário já citado na seção de caracterização do perfil do participante. Para o estudo de observação, foram disponibilizados apenas cinco avaliadores, ou seja, não houve escolha dos mesmos, já que todos foram utilizados na avaliação.

#### 6.2.2.6 Validade Interna do Estudo

De acordo com Pereira (2007), "A validade interna do estudo é definida como a capacidade de se repetir o comportamento do estudo atual com os mesmos participantes e objetos com que ele foi realizado". Desta forma, a validação do *Miningflow* garante um bom nível de validação interna, uma vez que podemos contar com os mesmos participantes e a mesma gama de serviços implementados para que os experimentos possam ser realizados novamente. Um fator que pode influenciar os experimentos é o fato de um participante "ajudar" o outro, fornecendo dicas de tarefas que já tenha realizado. Devido a esse problema é requisitado a cada participante do estudo que não forneça informações a outros participantes de maneira nenhuma.

#### 6.2.2.7 Validade Externa do Estudo

A validade externa do estudo se refere à aplicação do experimento em outros grupos diferentes do que foi utilizado. Desta forma, podemos ter problemas quanto à aplicação da proposta em outros grupos. Por se tratar de uma proposta

fundamentalmente acadêmica e científica, acabamos restringindo a gama de indivíduos que possam validar, experimentar e avaliar o sistema.

Entretanto, como este estudo visa avaliar a proposta num âmbito menor (utilização dentro da universidade e/ou em áreas de pesquisa bem definidas do domínio de mineração de textos) podemos considerar a validade externa do estudo suficiente.

### 6.2.3 Execução do Estudo de Observação

#### 6.2.3.1 Seleção dos Participantes

Para o estudo de observação foram disponibilizados cinco participantes. Estes participantes são alunos de mestrado do PESC da COPPE/UFRJ. Os participantes do experimento foram devidamente analisados de acordo com as restrições impostas na fase de planejamento do estudo de observação. Neste primeiro estudo de observação, selecionamos também participantes com pouca experiência em mineração de textos para podermos avaliar o quanto a ferramenta ajuda na definição do processo. Os participantes do estudo de observação foram escolhidos por métodos não-aleatórios (cinco pessoas foram convidadas para participar deste primeiro estudo de observação).

#### 6.2.3.2 Instrumentação

Os participantes do estudo de observação deveriam executar as três tarefas propostas no planejamento (definição do *workflow* abstrato, definição do *workflow* concreto e redefinição do *workflow*). Estas tarefas seriam executadas no portal *Miningflow*.

Ao se registrar no sistema o usuário tem a opção de escolher uma das três funcionalidades a serem avaliadas. A primeira opção a ser escolhida é a definição do *workflow* abstrato. Nesta etapa o usuário deve escolher o número de tarefas da sua cadeia de execução, e quais as classes da ontologia que definem cada etapa.

Executada esta etapa de definição do *workflow* abstrato, o usuário deve selecionar a opção de definição do *workflow* concreto. Esta opção carrega um *workflow* abstrato previamente gerado e disponibiliza serviços *web* previamente registrados em uma biblioteca de serviços da ferramenta. Para cada etapa do *workflow* será disponibilizada uma série de serviços passiveis de serem escolhidos de acordo com o

conceito que foi definido no *workflow* abstrato. Por exemplo, se no *workflow* abstrato foi definido que uma atividade do *workflow* era uma **tarefa de mineração de textos**, apenas serviços que executem uma tarefa de mineração de textos serão disponibilizados para o usuário.

Definido o *workflow* concreto, o usuário está apto a redefinir seu *workflow*. Desta forma ele pode carregar em memória um *workflow* concreto já definido e apenas trocar parâmetros de entrada, sem alterar a seqüência lógica de execução das tarefas.

#### 6.2.3.3 Procedimento de Participação

Nesta dissertação, existe apenas um estudo de participação já que todos os alunos/participantes executam as mesmas tarefas dentro do sistema. A seguir definimos todas as etapas do processo de participação:

1. Foi criada uma conta no servidor para cada participante do estudo.
2. O participante recebe treinamento presencial dado pelo experimentador.
3. O participante acessa o portal do *Miningflow*.
4. O participante define o seu *workflow* abstrato.
5. O participante define o seu *workflow* concreto.
6. O participante redefine seu *workflow* concreto.
7. O participante conclui suas tarefas no portal do *Miningflow*.
8. É entregue ao participante um questionário para avaliação qualitativa do *Miningflow*.
9. O participante preenche o formulário, respondendo as perguntas presentes no anexo B desta dissertação.

#### 6.2.3.4 Execução

Neste estudo de observação, os participantes não foram divididos em grupos. Desta maneira, todos os participantes receberam as mesmas tarefas para serem

executadas. Cada participante deveria executar as ações 4, 5 e 6 do tópico 6.2.3.3 e verificar se os dados armazenados correspondem com a realidade dos experimentos que foram definidos.

**Tabela 6.1 – Informações detalhadas sobre os participantes do estudo de observação**

| ID | Tarefas Executadas | Experiência com *Workflows* | Experiência com Mineração de Textos |
|---|---|---|---|
| 1 | 4, 5 e 6 | Pouco experiente | Experiente |
| 2 | 4, 5 e 6 | Experiência razoável | Pouco Experiente |
| 3 | 4, 5 e 6 | Experiente | Pouco Experiente |
| 4 | 4, 5 e 6 | Experiência razoável | Pouco experiente |
| 5 | 4,5 e 6 | Pouco experiente | Experiência razoável |

Através das respostas dadas pelos próprios participantes, pudemos perceber que 20% dos mesmos se consideram experientes no que se refere a *workflows* científicos, 40% consideravam sua experiência razoável e outros 40% se consideravam inexperientes já que haviam trabalhado muito pouco, ou nada, com este tipo de tecnologia.

No que se refere ao conhecimento do processo de mineração de textos, pudemos perceber que 20% dos mesmos se consideram experientes, outros 20% consideravam sua experiência razoável e 60% se consideravam inexperientes já que haviam trabalhado muito pouco, ou nada, com este tipo de tecnologia.

### 6.2.4 Análise dos Resultados do Estudo de Observação

Como nosso estudo de observação focou na análise qualitativa dos resultados, esta subseção exibe os resultados obtidos conforme o preenchimento do formulário de

avaliação qualitativa do *Miningflow*. Cada sub-tópico representa o que foi obtido com o questionário de avaliação da proposta.

Em relação à utilização de tecnologias de *workflows* científicos no domínio de mineração de textos, 80% dos entrevistados responderam que esta utilização pode ser extremamente útil enquanto 20% afirmou que a utilização destas técnicas se mostra útil (a graduação das respostas foi convencionada de 1 (inútil) até 5 (extremamente útil)).

Apesar de muitos dos usuários não possuírem conhecimentos profundos em *workflows* científicos, a maioria deles, após a aula de nivelamento que foi ministrada, concordou que muitos dos tópicos aplicados em tecnologias de *workflows* podem ser facilmente adaptados e utilizados no domínio de mineração de textos.

Em relação à criação de um *workflow* abstrato que servirá de base para a definição de experimentos (*workflows* concretos), 60% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo enquanto que outros 40% opinaram que esta funcionalidade eventualmente auxiliará no processo.

Os participantes que pertencem aos 40% que responderam que a funcionalidade será eventualmente importante (dois de cinco no total) são usuários atuais de sistemas que trabalham com mineração de dados, como o WEKA. Desta forma, eles já estão amplamente familiarizados com este tipo de tecnologia e com as implementações que já existem. Assim sendo, eles não vêem a necessidade (ou pelo menos não uma grande necessidade) de serem guiados durante o processo de definição. Já os 60% que responderam que esta funcionalidade sempre auxiliará no processo não são experientes em mineração de textos, e viram nesta funcionalidade um facilitador interessante para que eles possam definir um experimento.

Em relação à criação de um *workflow* concreto que definirá a seqüência de programas ou serviços *web* que serão efetivamente executados pela máquina de *workflows*, 100% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, uma vez que a mesma é a o ponto-chave do processo já que sem o *workflow* concreto, o experimento nunca poderá ser executado, uma vez que o mesmo é a tradução de um experimento em termos de programas ou serviços executáveis.

Em relação ao armazenamento dos dados de proveniência do experimento, 40% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, outros 40% responderam que eventualmente o armazenamento de dados será importante e 20% dos entrevistados responderam que nunca farão uso desta funcionalidade.

Neste caso, um dos usuários (20%) trabalha sozinho com mineração de dados e textos e não viu necessidade de registrar dados de proveniência no sistema. Como esta pessoa trabalha sozinha, todos os arquivos de saída e /ou resultados intermediários são catalogados por ela separadamente, com nomes de arquivos controlados. Em ambientes como este (apenas uma pessoa trabalhando) este modo de trabalho é plausível (apesar de não garantir que não haverá falhas), porém, quando se trabalha em equipe (e particularmente com equipes grandes), um usuário não tem conhecimento do que os outros estão fazendo, desta forma, este tipo de controle de torna inviável. Como esta resposta deste usuário refletia uma situação muito particular, não impactou em nenhuma decisão para modificar o *Miningflow*.

Em relação à utilização de ontologias para representar conceitos e auxiliar no processo de definição do experimento, 80% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo e 20% dos entrevistados responderam que esta funcionalidade eventualmente será importante.

Esta pergunta acaba ficando muito relacionada com a criação do *workflow* abstrato. Como na definição do *workflow* abstrato, a utilização de ontologias se faz o tempo todo presente, as pessoas que responderam que um *workflow* abstrato é sempre importante acabaram repetindo sua opinião aqui. Entretanto, um entrevistado que havia respondido que um *workflow* abstrato é eventualmente importante mudou de opinião neste quesito, pois analisou que a utilização de ontologias vai alem da criação de um *workflow* abstrato, já que a mesma fornece semântica para os dados que são armazenados, o que ele considerou importante e útil.

Em relação á verificação semântica realizada no momento das definições abstratas e concretas, apenas 20% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, enquanto que 60% dos entrevistados responderam que esta funcionalidade eventualmente será importante e outros 20% responderam que nunca farão uso desta funcionalidade.

Mais uma vez o usuário que respondeu que nunca fará utilização desta funcionalidade visualizou apenas o seu ambiente de trabalho (trabalhando sozinho) onde o mesmo conhece todos os serviços e /ou programas e possui controle sobre tudo. Quando lidamos com equipes maiores onde cada integrante desenvolve seus próprios serviços, o conhecimento fica disperso. Desta maneira, fica complicado para cada integrante conhecer os parâmetros de entrada e saída de cada serviço do *workflow*. Assim sendo, é importante para o sistema que esta verificação seja realizada de modo a diminuir erros. Dos 60% que responderam que eventualmente esta funcionalidade ajudará, o argumento foi que este tipo de funcionalidade acabará ajudando no início da utilização, mas os serviços que cada pessoa utilizará durante o decorrer do tempo não irão variar muito, assim sendo, depois de um tempo o usuário já conhecerá os serviços que são compatíveis sem auxílio do sistema.

Em relação à independência da ferramenta aos SGWf existentes, 100% dos entrevistados responderam que esta característica é uma das grandes vantagens da ferramenta, uma vez que não fica atrelada a nenhuma implementação. Como os SGWf não são padronizados, ou seja, a cada dia surgem novos, os existentes mudam e desaparecem alguns, não podemos ficar com experimentos "presos" a uma determinada implementação de SGWf.

Em relação à facilidade de uso do sistema, 80% dos entrevistados responderam que o sistema é fácil de ser manipulado enquanto que 20% dos entrevistados responderam que o sistema se mostra extremamente fácil em sua utilização. Isto nos mostra que a preocupação em desenvolver uma ferramenta de fácil utilização foi atendida ao menos em uma primeira versão, já que melhorias podem ser realizadas no protótipo.

Na avaliação final da ferramenta, 80% dos entrevistados apontaram que a proposta sempre atende o objetivo proposto que é oferecer suporte no processo de definição e execução de um experimento de mineração de textos, enquanto que 20% dos entrevistados responderam que eventualmente a ferramenta atende aos objetivos traçados. Destes 20% que responderam que a proposta atende eventualmente a alegação foi que existem alguns pontos necessários em experimentos de mineração de textos que não foram cobertos, como definir um *workflow* concreto apenas uma vez e automaticamente o sistema variar parâmetros de entrada em uma faixa de valores, por

exemplo. Atualmente no *Miningflow* para variar parâmetros, o usuário deve redefinir um *workflow* concreto para que possa informar novos valores de entrada.

Algumas observações foram realizadas nos questionários de avaliação. A principal delas foi em relação à interface ainda ser textual, uma vez que se torna um pouco menos intuitiva do que uma interface gráfica. Este problema já havia sido percebido, mas como a implementação do sistema visa avaliar uma proposta, focamos na implementação de funcionalidades essenciais para a avaliação. Uma interface gráfica pode ajudar o usuário nas tarefas, mas não inviabilizaria a execução do estudo de caso, além de demandar um esforço grande de implementação.

Outra observação que foi realizada foi a falta de ferramentas para realizar a carga ou para baixar informações do servidor onde os serviços irão ser executados. Além disso, foi requisitado um módulo para acesso direto a linha de comando do servidor, de modo que a máquina de *workflow* possa ser executada sem ter a necessidade de ser instalada na máquina do usuário. A criação de uma página para carga e para baixar informações do servidor, bem como uma área para invocação remota de comandos, foram incorporados ao sistema após o estudo de observação para que já pudessem ser utilizadas no momento do estudo de caso.

## 6.3 Estudo de Caso

Seguindo a proposta da definição de um processo de avaliação definida por Pereira (2007) baseado em Travassos (2005), o estudo de caso foi proposto e realizado em um ambiente não controlado, com apenas um servidor e participantes com perfis distintos e especialistas no domínio de mineração de textos. Este experimento não foi supervisionado diretamente pelo experimentador. Neste caso, o experimentador apenas acompanhou os usuários de forma que pudesse esclarecer eventuais dúvidas.

A definição deste estudo experimental segue o mesmo padrão do estudo de observação que foi realizado anteriormente com usuários não especialistas e especialistas no domínio de mineração de textos. A execução deste estudo pode ser tratada como uma maneira de comprovar que o estudo de observação foi válido e que a proposta foi validada em um ambiente não controlado, sem supervisão direta do experimentador e com usuários com conhecimentos variados sobre o assunto. Os

participantes do estudo de caso serão divididos entre alunos de mestrado e doutorado do PESC e do PEC da COPPE/UFRJ.

Além disto, o estudo de caso, já se beneficiou de muitas das melhorias propostas no estudo de observação de forma que os usuários pudessem avaliar uma ferramenta o mais completa possível dentro dos objetivos traçados.

### 6.3.1 Contexto

**Analisar** a utilização do sistema *Miningflow*; **com o propósito de** avaliar a viabilidade de sua utilização na geração de *workflows* (concreto e abstrato) **aplicados no** domínio de mineração de texto; **referente** a satisfação do usuário **do ponto de vista do** aprendiz e do especialista.

#### 6.3.1.1 Perfil dos Participantes

Assim como no estudo de observação, os participantes acessaram o sistema através de uma plataforma *web*. Deste modo, era necessário que os participantes tivessem conhecimentos básicos sobre acesso à *web*, quesito que é facilmente contemplado uma vez que trabalhamos com indivíduos da área de ciência da computação.

É necessário também que os participantes tenham algum conhecimento sobre o domínio da aplicação e *workflows* científicos, Caso contrário, pode ser necessário que alguns aspectos do processo sejam esclarecidos (principalmente em relação a *workflows* científicos, uma vez que para o estudo de caso, os usuários serão ao menos razoavelmente experientes no domínio de mineração de textos) para que esses participantes usem de forma apropriada às funcionalidades do sistema.

O estudo de caso contou com a participação de oito voluntários, dentre eles, profissionais da área de computação (graduandos e mestres), e alunos de mestrado e doutorado do PESC e do PEC da COPPE/UFRJ.

#### 6.3.1.2 Recursos Utilizados

Para que o experimento pudesse ser realizado, foi disponibilizado um computador Pentium Core II Duo 3.4 Ghz com 2 Gb de Memória RAM e HD de 500

Gb conectado a *internet* através de uma conexão a cabo de 2Mbit e funcionando como o servidor da aplicação e de banco de dados para o *Miningflow*.

Além do acesso à ferramenta, foram disponibilizados os mesmos dois questionários utilizados no experimento de observação impressos para os participantes do experimento:

1. Um questionário para qualificação dos participantes constante do Anexo A desta dissertação;

2. Um questionário para avaliação qualitativa do *Miningflow* constante do anexo B desta dissertação.

Assim como no estudo de observação, a experiência tanto em relação à mineração de texto quanto em relação às tecnologias de *workflows* serão avaliadas pelo questionário antes do início do estudo de caso. Após o término do experimento, é solicitado que os participantes respondam o questionário elaborado para avaliação da ferramenta. As questões foram definidas considerando as características específicas do processo de mineração de textos em conjunto com as técnicas de *workflows* e o uso de ontologias.

Nas seções que se seguem, será apresentada a descrição do estudo de caso elaborado. A descrição está dividida em quatro partes principais a serem avaliadas: a primeira visa a definição de um *workflow* abstrato (*template*) que será utilizado para guiar usuários teoricamente menos experientes. A segunda parte visa a definição do *workflow* concreto e a terceira a redefinição de um *workflow* concreto, habilitando a mudança de parâmetros do *workflow*. A quarta parte do estudo de caso visa recuperar e avaliar dados de proveniência que foram captados e armazenados na base de dados durante a execução do experimento.

### 6.3.1.3 Projeto Piloto

Como o protótipo não foi radicalmente alterado e o ambiente se manteve o mesmo (ambiente *web*) não se mostrou necessária a realização de um novo projeto piloto para avaliação prévia da ferramenta.

### 6.3.1.4 Treinamento dos Usuários

Diferentemente do estudo de observação, o treinamento dos usuários para o estudo de observação não será realizado presencialmente e ministrado pelo avaliador, e sim, através de um pequeno texto explicando os passos que devem ser realizados.

Numa visão geral, o treinamento será composto de quatro rodadas (que seguem as fases do planejamento do experimento): geração do *workflow* abstrato, geração do *workflow* concreto, redefinição do *workflow* e avaliação dos resultados.

## 6.3.2 Planejamento do experimento

O experimento será composto de duas rodadas com quatro partes cada uma. Na primeira rodada, o experimento será composto de quatro partes principais: definição do *workflow* abstrato que refletirá o processo clássico de mineração de textos (pré-processamento, mineração e pós-processamento), definição do *workflow* concreto e redefinição de um *workflow* concreto. Nesta rodada, o usuário poderá escolher qual tipo de mineração deseja realizar (classificação, agrupamento, etc.). Após executar o experimento, o usuário deve recuperar dados armazenados e avaliá-los para que se possam testar os mecanismos de proveniência disponibilizados.

Na segunda rodada o experimento também será composto de quatro partes principais: definição do *workflow* abstrato que refletirá apenas a etapa de pré-processamento do processo, definição do *workflow* concreto e redefinição de um *workflow* concreto. Nesta rodada, o usuário poderá escolher apenas quais as atividades de pré-processamento ele deseja executar. Optamos por realizar um teste somente da etapa de pré-processamento, pois esta se mostra a mais custosa do processo inteiro e é a que contém a maior quantidade de sub-etapas que podem variar de experimento para experimento. Após executar o experimento, o usuário deve recuperar dados armazenados e avaliá-los para que se possam testar os mecanismos de proveniência disponibilizados.

Cada participante recebeu informações genéricas sobre as tarefas a serem executadas de forma que pudessem avaliar o quanto o sistema ajudou na geração de um *workflow*. As sub-seções 6.3.2.1 6.3.2.2 6.3.2.3 e 6.3.2.4 explicam detalhadamente as tarefas que foram executadas pelos avaliadores.

#### 6.3.2.1 Parte I: Definição do *Workflow* Abstrato

Cada participante gerará um *workflow* abstrato que será usado como base para a geração dos *workflows* concretos a seguir. O participante terá que definir a quantidade de tarefas que irão compor o *workflow* (na primeira rodada serão obrigatoriamente três atividades e na segunda rodada a quantidade de atividades pode variar de acordo com a quantidade de sub-etapas de pré-processamento que o usuário deseja executar) e quais as classes da ontologia que estarão associadas a cada tarefa.

#### 6.3.2.2 Parte II: Definição do *Workflow* Concreto

Cada participante utilizará o *workflow* abstrato gerado previamente para selecionar os serviços a serem executados e definir assim o seu *workflow* em termos de serviços *web*, definindo assim o seu experimento.

#### 6.3.2.3 Parte III: Redefinição do *Workflow* Concreto

Cada participante recuperará um *workflow* concreto gerado e redefinirá seus parâmetros, gerando uma nova versão do *workflow*. É importante ressaltar que nesta fase, o participante apenas redefinirá parâmetros, sem ter que escolher serviços e/ ou operações a serem executadas.

#### 6.3.2.4 Parte IV: Recuperação e Análise dos Dados Armazenados

Cada participante recuperará os dados armazenados relativos à execução de um determinado *workflow* concreto, de forma que possa ser realizada uma análise crítica dos resultados e avaliar se o experimento foi bem sucedido ou não. Esta fase é importante, pois além de validar o experimento, podemos avaliar os mecanismos de captura e recuperação de dados de proveniência.

#### 6.3.2.5 Critérios de Avaliação

Para avaliar a proposta *Miningflow* se faz necessário que avaliemos os ganhos obtidos pelos usuários do sistema e também as dificuldades encontradas pelos mesmos. Para que pudéssemos realizar esta avaliação, utilizaremos o processo de avaliação qualitativa do *Miningflow*, já proposto anteriormente no estudo de observação. A

satisfação dos participantes será avaliada de acordo com um questionário que será distribuído para os mesmos e que se encontra presente no anexo B desta dissertação.

#### 6.3.2.6 Validade Interna do Estudo

O *Miningflow* garante um bom nível de validação interna, uma vez que podemos contar com os mesmos participantes e a mesma gama de serviços implementados para que os experimentos possam ser realizados novamente. Um fator que poderia influenciar os experimentos é o fato de um participante "ajudar" o outro (o que poderia acontecer no estudo de observação), fornecendo dicas de tarefas que já tenha realizado. Como no estudo de caso os participantes se encontravam geograficamente dispersos este problema não poderia ocorrer.

#### 6.3.2.7 Validade Externa do Estudo

Assim como já definido no estudo de observação, a validade externa do estudo se refere a aplicação do experimento em outros grupos diferentes do que foi utilizado. Assim como no estudo de observação, podemos ter problemas quanto a aplicação da proposta em outros grupos. Por se tratar de uma proposta fundamentalmente acadêmica e científica, acabamos restringindo a gama de indivíduos que possam validar, experimentar e avaliar o sistema. Entretanto, como este estudo visa avaliar a proposta num âmbito menor (utilização dentro da universidade e/ou em áreas de pesquisa bem definidas) podemos considerar a validade externa do estudo suficiente.

### 6.3.3 Execução do Estudo de Observação

#### 6.3.3.1 Seleção dos Participantes

Para o estudo de observação foram selecionados oito participantes dentre a gama de voluntários que foi disponibilizada. Estes participantes são alunos de mestrado e doutorado do PESC e do PEC da COPPE/UFRJ. Os participantes do experimento foram devidamente analisados de acordo com as restrições impostas na fase de planejamento do estudo de caso. Para o estudo de caso, selecionamos participantes com experiência em mineração de textos. Os participantes do estudo de caso foram escolhidos por métodos aleatórios uma vez que contávamos com uma quantidade maior do que a estipulada de alunos para avaliar a ferramenta.

### 6.3.3.2 Instrumentação

Os participantes do estudo de caso deveriam executar as quatro tarefas propostas no planejamento (definição do *workflow* abstrato, definição do *workflow* concreto, redefinição do *workflow* e análise dos dados de proveniência) nas duas rodadas que foram definidas. Estas tarefas seriam executadas no portal *Miningflow*.

Ao se registrar no sistema o usuário tem a opção de escolher uma das quatro funcionalidades a serem avaliadas. A primeira opção a ser escolhida é a definição do *workflow* abstrato. Nesta etapa o usuário deve escolher o número de tarefas da sua cadeia de execução, e quais as classes da ontologia que definem cada etapa.

Executada esta etapa de definição do *workflow* abstrato, o usuário deve selecionar a opção de definição do *workflow* concreto. Esta opção carrega um *workflow* abstrato previamente gerado e disponibiliza serviços *web* previamente registrados em uma biblioteca de serviços da ferramenta. Para cada etapa do *workflow* será disponibilizada uma série de serviços passiveis de serem escolhidos de acordo com o conceito que foi definido no *workflow* abstrato. Por exemplo, se no *workflow* abstrato foi definido que uma atividade do *workflow* era uma **tarefa de mineração de textos**, apenas serviços que executem uma tarefa de mineração de textos serão disponibilizados para o usuário.

Definido o *workflow* concreto, o usuário está apto a redefinir seu *workflow*. Desta forma ele pode carregar em memória um *workflow* concreto já definido e apenas trocar parâmetros de entrada, sem alterar a seqüência lógica de execução das tarefas.

Após executar os *workflows* que foram redefinidos ou definidos, o usuário está apto a recuperar os dados de proveniência e avaliar o quão bom ou ruim foi o experimento em questão.

### 6.3.3.3 Procedimento de Participação

Nesta dissertação, existe apenas um roteiro de estudo de caso já que todos os alunos/ participantes executam as mesmas tarefas dentro do sistema, variando apenas os serviços que irão compor o experimento. A seguir definimos todas as etapas do processo de participação:

1. Foi criada uma conta no servidor para cada participante do estudo.
2. O participante recebe treinamento não presencial através de documento explicativo.
3. O participante acessa o portal do *Miningflow*.
4. O participante define o seu *workflow* abstrato (1).
5. O participante define o seu *workflow* concreto (1).
6. O participante redefine seu *workflow* concreto (1).
7. O participante recupera os dados de proveniência registrados e os avalia criteriosamente (1).
8. O participante define o seu *workflow* abstrato (2).
9. O participante define o seu *workflow* concreto (2).
10. O participante redefine seu *workflow* concreto (2).
11. O participante recupera os dados de proveniência registrados e os avalia criteriosamente (2).
12. O participante conclui suas tarefas no portal do *Miningflow*.
13. É entregue ao participante um questionário para avaliação qualitativa do *Miningflow*.
14. O participante preenche o formulário, respondendo as perguntas presentes no anexo B desta dissertação.

#### 6.3.3.4 Execução

No estudo de caso, os participantes não foram divididos em grupos. Desta maneira, todos os participantes receberam as mesmas tarefas para serem executadas. Cada participante deveria executar as ações 4, 5, 6, 7, 8, 9, 10 e 11 do tópico 6.3.3.3 .

Tabela 6.2 – Informações detalhadas sobre os participantes do estudo de caso

| ID | Tarefas Executadas | Experiência com *Workflows* | Experiência com Mineração de Textos |
|---|---|---|---|
| 1 | 4, 5, 6, 7, 8, 9, 10 e 11 | Experiência razoável | Pouco experiente |
| 2 | 4, 5, 6, 7, 8, 9, 10 e 11 | Pouco experiente | Experiência razoável |
| 3 | 4, 5, 6, 7, 8, 9, 10 e 11 | Pouco experiente | Experiente |
| 4 | 4, 5, 6, 7, 8, 9, 10 e 11 | Pouco experiente | Experiência razoável |
| 5 | 4, 5, 6, 7, 8, 9, 10 e 11 | Experiência razoável | Pouco experiente |
| 6 | 4, 5, 6, 7, 8, 9, 10 e 11 | Pouco Experiente | Experiência razoável |
| 7 | 4, 5, 6, 7, 8, 9, 10 e 11 | Experiente | Pouco experiente |
| 8 | 4, 5, 6, 7, 8, 9, 10 e 11 | Pouco experiente | Experiente |

### 6.3.3.5 Análise do Perfil dos Participantes do Estudo de Caso

Através das respostas dadas pelos próprios participantes, pudemos perceber que 13% dos mesmos se consideram experientes no que se refere a *workflows* científicos, 25% consideravam sua experiência razoável e outros 62% se consideravam inexperientes já que haviam trabalhado muito pouco, ou nada, com este tipo de tecnologia.

No que se refere ao conhecimento do processo de mineração de textos, pudemos perceber que 25% dos mesmos se consideram experientes, outros 38% consideravam sua experiência razoável e 37% se consideravam inexperientes já que haviam trabalhado muito pouco, ou nada, com este tipo de tecnologia.

### 6.3.4 Análise dos Resultados do Estudo de Caso

Assim como em nosso estudo de observação, que focou na análise qualitativa dos resultados, esta subseção exibirá os resultados obtidos conforme o preenchimento do formulário de avaliação qualitativa do *Miningflow* durante o estudo de caso que foi realizado. Cada sub-tópico abaixo descrito representará o que foi obtido com no questionário de avaliação da proposta.

Em relação à utilização de tecnologias de *workflows* científicos no domínio de mineração de textos, 75% dos entrevistados responderam que esta utilização pode ser extremamente útil enquanto 25% afirmaram que a utilização destas técnicas se mostra útil. A graduação das respostas foi convencionada de 1 (inútil) até 5 (extremamente útil).

Apesar de muitos dos usuários não possuírem conhecimentos profundos em *workflows* científicos, a maioria deles, após as devidas explicações sobre conceitos de *workflows* científicos dadas no texto explicativo, concordou que muitos dos tópicos aplicados em tecnologias de *workflows* podem ser facilmente adaptados e utilizados no domínio de mineração de textos, fazendo assim, com que o usuário final esteja apto a ter um ganho substancial no processo.

Em relação à criação de um *workflow* abstrato que servirá de base para a definição de experimentos (*workflows* concretos), 87% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo enquanto que outros 13% acham que esta funcionalidade eventualmente auxiliará no processo.

Da mesma maneira como pôde ser constatado no estudo de observação, os participantes que pertencem aos 13% que responderam que a funcionalidade será eventualmente importante são usuários que costumam trabalhar sozinhos e já possuem um conhecimento razoável do processo de mineração de textos Assim sendo, podem não ter vislumbrado as dificuldades que um analista de mineração de textos iniciante pode possuir no momento de definir o seu experimento. Entretanto, a grande parte dos entrevistados conseguiu verificar que, para um iniciante, muitas vezes o processo de definição pode ser muito complicado por não possuir o conhecimento do processo completamente concretizado.

Em relação à criação de um *workflow* concreto que definirá a seqüência de programas ou serviços *web* que serão efetivamente executados pela máquina de *workflows*, 100% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, uma vez que a mesma é a o cerne do processo já que o *workflow* concreto é uma representação do experimento em termos de programas ou serviços executáveis e, sem ele, o experimento nunca poderá ser executado.

Em relação ao armazenamento dos dados de proveniência do experimento, 74% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, outros 13% responderam que eventualmente o armazenamento de dados será importante e 13% dos entrevistados responderam que nunca farão uso desta funcionalidade.

Neste caso, assim como no estudo de observação, um dos usuários (13%) trabalha sozinho com mineração de dados e textos e pode não ter visto a necessidade de registrar dados de proveniência no sistema, já que todo controle de entrada, geração e saída de resultados é feito somente por ele. Como esta resposta deste usuário refletia uma situação muito particular, não impactou em nenhuma decisão para modificar o ambiente *Miningflow* ou que tornasse negativa a avaliação da ferramenta.

Em relação à utilização de ontologias para representar conceitos e auxiliar no processo de definição do experimento, 75% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo e 25% dos entrevistados responderam que esta funcionalidade eventualmente será importante.

Assim como no estudo de observação, esta pergunta está intimamente relacionada com a criação do *workflow* abstrato e análise dos dados de proveniência no sistema. Como na definição do *workflow* abstrato, a utilização de ontologias se faz muito presente, a maioria das pessoas que responderam que um *workflow* abstrato é sempre importante acabaram repetindo sua opinião aqui. Entretanto, um entrevistado que havia respondido que um *workflow* abstrato é sempre importante mudou de opinião neste quesito, pois argumentou que a utilização de um *workflow* abstrato pode ser realizada mesmo sem a utilização de ontologias e que a incorporação da ontologia no sistema gerou um complicador, uma vez que os dados da ontologia devem ser previamente carregados para o sistema para que possam ser utilizados.

A opinião deste usuário tem seus prós e contras e deve ser analisada. É indiscutível que o módulo de definição abstrata pode ser implementado sem a utilização de ontologias, bastando que dentro do próprio sistema exista uma categorização de fases, algoritmos, etc. Entretanto, a utilização de ontologias nos deixa a disposição um ferramental grande, além do uso de ontologias já ter se tornado um padrão no que tange a representação do conhecimento.

A incorporação da ontologia no sistema pode ter sim, gerado um complicador para o usuário, mas há de se lembrar que o usuário que carregará as informações da ontologia para o sistema é o usuário especialista e administrador, e este processo de carga dos dados se torna transparente para o restante dos usuários, sem implicar em nenhum tipo de trabalho adicional em suas tarefas.

Em relação à verificação semântica realizada no momento das definições abstratas e concretas, apenas 13% dos entrevistados responderam que esta funcionalidade sempre auxiliará no processo, enquanto que 74% dos entrevistados responderam esta funcionalidade eventualmente será importante e outros 13% responderam que nunca farão uso desta funcionalidade.

Assim como no estudo de observação, o usuário que respondeu que nunca fará utilização desta funcionalidade visualizou apenas o seu ambiente de trabalho (trabalhando sozinho) onde o mesmo conhece todos os serviços e/ ou programas e possui controle sobre tudo. Dos 74% que responderam que eventualmente esta funcionalidade ajudará, o argumento principal se manteve o mesmo: que este tipo de funcionalidade acabará ajudando no início, mas os serviços que cada pessoa utilizará não irão variar muito com o tempo, logo, depois de um intervalo de tempo o usuário já conhecerá os serviços que são compatíveis sem auxilio do sistema.

Em relação à independência da ferramenta aos SGWf existentes, 100% dos entrevistados responderam que esta característica é uma das grandes vantagens da ferramenta, uma vez que não fica atrelada a nenhuma implementação. Isso nos mostra que uma de nossas maiores preocupações (desenvolver uma ferramenta o mais independente de tecnologias) foi sanada e que os usuários souberam reconhecer uma das vantagens que consideramos primordiais na ferramenta.

Em relação à facilidade de uso do sistema, 87% dos entrevistados responderam que o sistema é fácil de ser manipulado enquanto que 13% dos entrevistados responderam que o sistema se mostra com uma complexidade média de ser manipulado. Esta informação nos mostra que apesar de ter sido projetado para ser o mais simples para o usuário final, o *Miningflow* ainda se mostra deficiente em certos pontos, uma vez que não possui uma interface 100% visual e amigável, onde o usuário ainda necessita interagir muito textualmente.

Em relação à avaliação final da ferramenta, 87% dos entrevistados apontaram que a proposta sempre atende o objetivo proposto que é oferecer suporte no processo de definição e execução de um experimento de mineração de textos, enquanto que 13% dos entrevistados responderam que eventualmente a ferramenta atende aos objetivos traçados. Destes 13% que responderam que a proposta atende eventualmente, o argumento utilizado foi que ainda existem alguns pontos necessários em experimentos de mineração de textos que não foram cobertos, como definir um *workflow* concreto apenas uma vez e automaticamente o sistema variar parâmetros de entrada em uma faixa de valores, exibir o estado de um experimento (se já foi executado ou não) ou ainda a reutilização de dados intermediários em execuções futuras, por exemplo.

Algumas observações foram realizadas nos questionários de avaliação. A principal delas foi em relação à interface ainda ser textual, uma vez que se torna um pouco menos intuitiva do que uma interface gráfica. Este problema já havia sido percebido durante a implementação e no estudo de observação, mas como a implementação do sistema visa avaliar uma proposta, focamos na implementação de funcionalidades essenciais para a avaliação. Uma interface gráfica pode ajudar o usuário nas tarefas, mas não inviabilizaria a execução do estudo de caso, além de demandar um esforço grande de implementação. Outras observações importantes foram realizadas, dentre elas podemos destacar um agendamento de execução de experimentos pelos usuários. O *Miningflow* permitiria que o usuário agendasse uma execução do seu experimento e automaticamente o sistema invocasse a máquina de *workflow* no dia e horário estipulados. Além disto, foi requisitado que o sistema exiba o estado de um experimento, mostrando se o mesmo foi executado ou não.

A Tabela 6.3 nos mostra alguns dos resultados adicionais que foram obtidos durante a validação do *Miningflow*, como por exemplo os percentuais atribuídos nas escalas de graduação por módulo implementado do *Miningflow*.

**Tabela 6.3 – Resultados obtidos com o estudo de caso**

| **O quão fácil ou difícil você achou usar o *Miningflow*?** | | | | |
|---|---|---|---|---|
| Extremamente Difícil **0%** | **0%** | **13,5%** | **87,5%** | Extremamente Fácil **0%** |
| **Você considera que o *Miningflow* atende o objetivo proposto que é dar suporte ao processo de desenvolvimento de novas aplicações de mineração de texto através da modelagem do processo como um *workflow* científico?** | | | | |
| Sempre **87,5%** | Eventualmente **12,5%** | | Nunca **0%** | |
| Qualifique as funcionalidades abaixo | | | | |
| **Controle de Acesso** | | | | |
| Extremamente Inútil **0%** | **0%** | **37,5%** | **62,5%** | Extremamente Útil **0%** |
| **Definição do *Workflow* Abstrato** | | | | |
| Extremamente Inútil **0%** | **0%** | **0%** | **12,5%** | Extremamente Útil **87,5%** |
| **Definição do *Workflow* Concreto** | | | | |
| Extremamente Inútil **0%** | **0%** | **0%** | **12,5%** | Extremamente Útil **87,5%** |
| **Re-definição do *Workflow* Concreto** | | | | |
| Extremamente Inútil **0%** | **0%** | **0%** | **12,5%** | Extremamente Útil **87,5** |
| **Análise dos Dados de Proveniência** | | | | |
| Extremamente Inútil **0%** | **0%** | **0%** | **50%** | Extremamente Útil **50%** |
| **Ontologia de Mineração de Textos** | | | | |
| Extremamente Inútil **0%** | **0%** | **0%** | **12,5%** | Extremamente Útil **87,5%** |
| **Você utilizaria o *Miningflow* como forma de apoiar o desenvolvimento de experimentos de mineração de textos?** | | | | |
| Sim **100%** | | | Não 0% | |
| **Você recomendaria a utilização do *Miningflow* como forma de apoiar desenvolvimento de experimentos de mineração de textos?** | | | | |
| Sim **100%** | | | Não **0%** | |

As Figuras 6.1 e 6.2 nos mostram as avaliações dos usuários quanto à importância de cada módulo dentro da arquitetura. Foi requisitado a cada usuário que numerasse em ordem de importância cada módulo do sistema. A figura 6.1 nos mostra as opiniões dos usuários individualmente com a graduação de cada módulo em separado. A figura 6.2 nos mostra um gráfico contendo a média simples das notas

definidas pelos usuários para cada módulo, mostrando assim o grau de importância médio de cada módulo da arquitetura.

Pelo que pode ser observado, os usuários consideram mais importantes os módulos de definição concreta e definição abstrata. O módulo de definição concreta, mesmo que implementado de outras formas, pode ser encontrado em vários SGWf existentes (Taverna, Kepler, VisTrails), entretanto, o módulo de definição do *workflow* abstrato pode ser considerado um diferencial na proposta. Dos SGWf analisados, nenhum disponibilizava a opção de definir um experimento em termos de conceitos (em mais alto nível), e como foi mencionado pelos avaliadores, esta funcionalidade se torna muito útil principalmente porque agrega semântica ao experimento e auxilia os usuários menos experientes. Logo a seguir o módulo de redefinição foi considerado mais importante, já que é um grande facilitador para o experimentador, uma vez que poupa tempo do mesmo e evita re-trabalho. A utilização de ontologias, o registro e a análise de dados de proveniência foram classificados logo em seguida, e com uma graduação muito semelhante. É importante ressaltar que, apesar dos usuários classificarem a utilização de ontologias com uma graduação menor que as dos módulos de definição, a ontologia está subjacente ao módulo de definição abstrata, assim sendo, indiretamente esta foi classificada no topo. Por último, o módulo de controle de acesso foi considerado o menos importante pelos usuários, mesmo porque é um módulo de infra-estrutura, não fazendo parte da proposta em si.

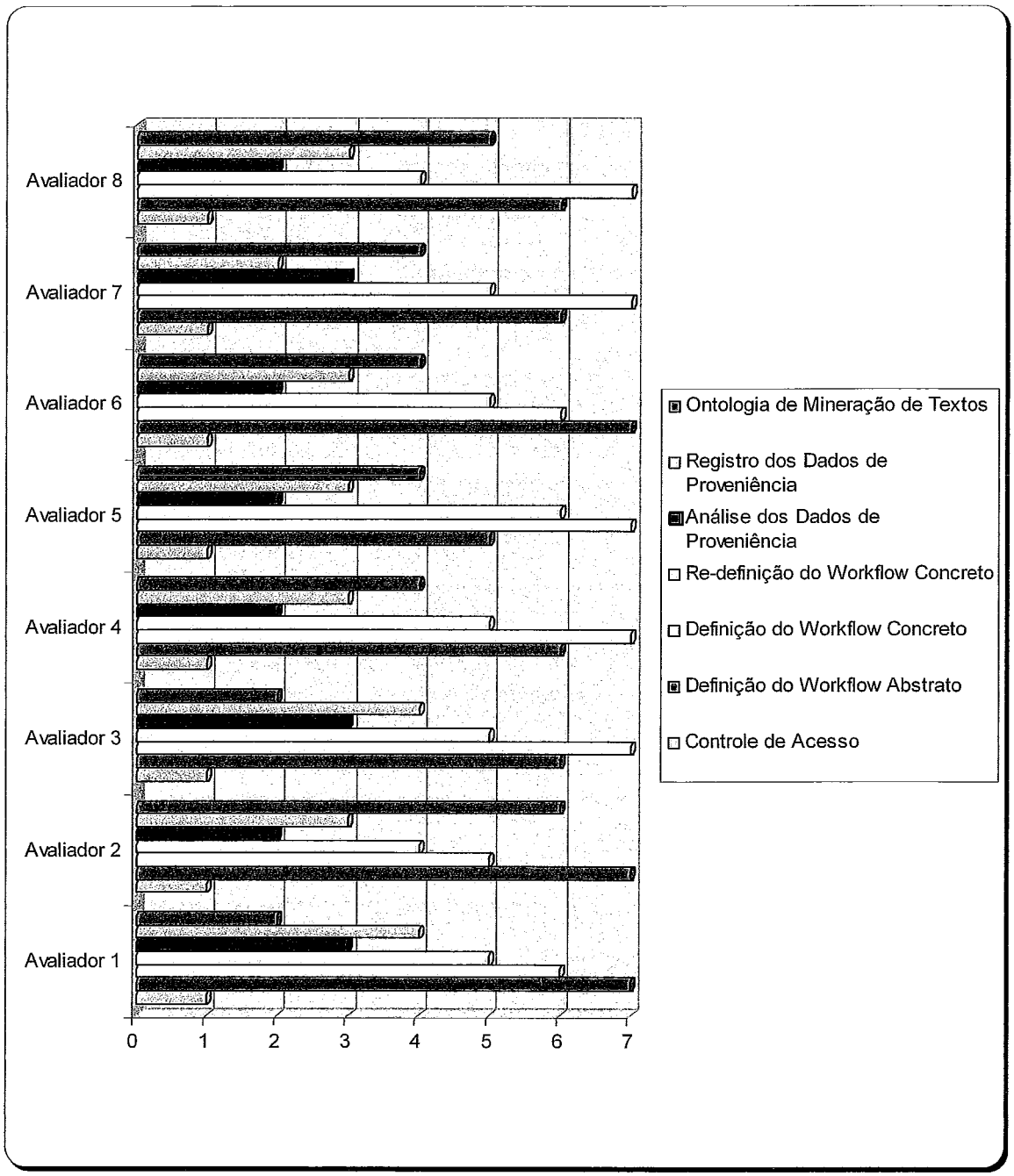


**Figura 6.1 – Graduação dos módulos quanto a sua importância.**

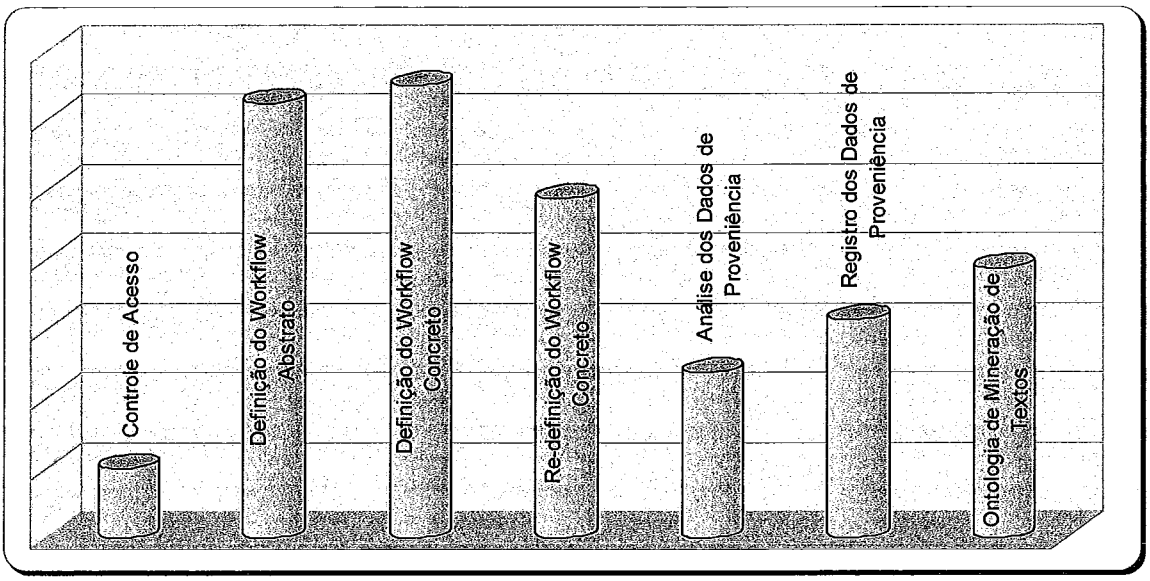


**Figura 6.2 –Média da graduação dos módulos quanto a sua importância.**

## 6.4 Conclusão

A partir da compilação dos resultados colhidos durante as validações, podemos realizar uma série de avaliações sobre o *Miningflow* a fim de gerar uma conclusão e uma avaliação final da proposta.

Em todas as avaliações dos usuários menos experientes (três), foi ressaltado que o módulo de definição abstrata oferece um mecanismo útil no que se refere ao apoio no desenvolvimento de novas aplicações. O Argumento utilizado por todos foi que quando tratamos de *workflows* abstratos simples (apenas com as três fases clássicas de mineração de textos, por exemplo), não podemos visualizar o problema. Mas, como um usuário ressaltou durante a avaliação, se quisermos criar experimentos detalhados em nível de funções de pré-processamento, por exemplo, o *workflow* pode não ser tão trivial quanto pode parecer. Conhecer quais funções irão compor o processo requer uma expertise que nem todos os usuários possuem. Além disso, foi considerado importante por todos os usuários a "sugestão" automática dos serviços *web* na hora de criar um *workflow* concreto. Conforme foi mencionado pelos avaliadores e também como já mencionamos nesta dissertação, conforme a equipe de desenvolvimento for aumentando, a quantidade de serviços também aumentará e em pouco tempo não haverá como se conhecer todos os serviços existentes e nem a qual fase do processo cada um está associado.

Outro ponto que foi ressaltado pelos usuários menos experientes (três) e por alguns com certa experiência (dois) no processo foi em relação à validação semântica que o *Miningflow* executa. Como em equipes muito grandes de analistas fica impossível conhecer todas as implementações que são realizadas, não há como saber quais implementações são incompatíveis entre si. Dado este ponto, foi considerado por muitos usuários (cinco) um mecanismos de suma importância a validação de compatibilidade entre as entradas e as saídas dos serviços que compõem o *workflow*.

Uma destas avaliações que pode ser realizada é em relação aos dados de proveniência e sua captura e análise. Apesar de metade dos avaliadores considerarem o módulo de análise de dados de proveniência extremamente importante e a outra metade importante, algumas críticas e avaliações foram realizadas em relação a este ponto. No sistema foram disponibilizados apenas três níveis de armazenamento dos dados de proveniência: Sem proveniência, total e parcial. A proveniência total armazena todos os resultados intermediários, a hora e data em que foram gerados e por qual serviço. A proveniência parcial armazena os mesmos dados, mas somente para a entrada e saída do experimento, sem se importar com os dados intermediários.

Foi requisitada pelos usuários a opção de escolha de quais dados seriam armazenados, bem como de incluir anotações nos dados. Além disto, foi argumentado que os serviços de proveniência existentes no *Miningflow* não lidam com todos os tipos de dados, apenas com os tipos já explicados no capítulo 5 desta dissertação. Para que o sistema fosse completo, segundo eles, o sistema deveria estar apto a trabalhar com todos os tipos de dados.

Como o objetivo da implementação do módulo de consulta de proveniência foi o de prova de conceito e não o de contribuição na definição de esquemas de proveniência nem tampouco modelos de captura e recuperação de dados de proveniência, mesmo que esta avaliação fosse negativa, não geraria um impacto muito grande na avaliação final da proposta. Entretanto, apesar das limitações, a proposta do módulo e a interface de recuperação dos dados foi considerada satisfatória por todos os usuários que realizaram a análise.

Outro ponto mencionado pelos usuários na avaliação foi em relação à execução do experimento. Foi incorporada na ferramenta uma opção de execução remota do

experimento, ou seja, o usuário pode executar seu experimento no próprio servidor, sem a necessidade de possuir uma máquina de *workflow* instalada em seu computador. Entretanto, para que isto seja possível, o *workflow* não pode gerar saídas gráficas (já que as saídas gráficas necessitam da interface atrelada à máquina de *workflow* para ser exibida). Mas, apesar disto, o propósito do mesmo foi avaliado como uma contribuição efetiva da ferramenta, pois faz com que nenhuma instalação ou execução seja realizada na máquina do usuário. Os dados são carregados para uma área específica do servidor, o experimento é executado remotamente e os resultados são capturados para análise.

A partir destas análises sobre as avaliações, podemos concluir que, apesar de possuir algumas deficiências, o *Miningflow* atingiu o seu objetivo primordial que foi oferecer um ambiente que disponibilizasse meios de apoiar o analista durante as etapas do ciclo de vida de um experimento de mineração de textos. Abaixo são listadas as etapas do ciclo de vida e o módulo relacionado a cada uma delas:

**Definição**: através dos módulos de definição abstrata, definição concreta e redefinição. Estes módulos foram bem avaliados pelos usuários e as observações realizadas (muitas das quais mostradas anteriormente nesta mesma seção) nos mostraram que oferecem o apoio necessário aos usuários menos experientes e aos experientes.

**Execução**: as funcionalidades de carga de arquivos, baixa de arquivos e de execução remota foram classificadas como interessantes e úteis pelos usuários, já que evitam que os mesmos necessitem ter uma máquina de *workflow* local, alem de repassar todo o processamento para o servidor.

**Análise e documentação dos resultados**: esta etapa se dá através das consultas pré-definidas que o sistema disponibiliza baseado nos dados de proveniência capturados, e apesar de limitada, foi bem avaliada pelos usuários.

Além destes pontos, outro tópico elogiado por grande parte dos usuários (sete) foi o fato de o Miningflow ser uma ferramenta de código aberto e não atrelada a nenhuma tecnologia (proprietária ou não).

Muitas das observações realizadas durante o estudo de caso e de observação são pertinentes e serão levadas em consideração no aprimoramento imediato da ferramenta ou serão incorporadas como trabalhos futuros de novas versões da mesma. É importante

ressaltar que não foram sugeridas modificações do ponto de vista do apoio semântico, por exemplo. As sugestões foram operacionais da ferramenta em si e não do modelo de apoio ao ciclo de vida de um experimento de mineração de textos.

# Capítulo 7 Conclusão

A área de mineração de textos utiliza muitas das técnicas já consagradas da área de mineração de dados agregadas a técnicas de busca e recuperação da informação, para desenvolver novas soluções para problemas que envolvem o manuseio de grandes coleções de textos escritos em linguagem natural com o objetivo de extrair conhecimento intrínseco desta massa de dados. Estas técnicas se referem à captação de textos em linguagem natural, processamento e transformação da informação processada em estruturas manipuláveis para apresentá-las em uma estrutura formal.

Considerando a necessidade premente de uma solução para oferecer apoio ao analista de mineração de textos durante o ciclo de vida de seu experimento (definição, execução e análise e documentação de resultados) essa dissertação propõe o sistema MiningFlow. A solução apresentada visa atender as especificidades do domínio da mineração de textos através do uso de uma ontologia de domínio aproveitando as vantagens dos SGWf existentes, ao mesmo tempo em que não cria dependência desses sistemas.

*MiningFlow* proporciona o uso combinado da gerência de experimentos científicos (apoiados por tecnologias de *workflows*) com uma ontologia de domínio para mineração de textos. O *MiningFlow* atua como uma camada intermediária entre o usuário analista de mineração de textos e o SGWf, de forma a prover ferramentas com semântica para o registro e a descoberta de recursos (leia-se programas, serviços, *scripts*) durante o processo de mineração de textos, especificamente na composição das atividades do *workflow* (durante a modelagem de experimentos) e na sua execução.

Durante a implementação da MiningFlow, uma ontologia voltada para o domínio de mineração de textos (*MF-Ontology*) foi desenvolvida, baseando-se na proposta de Cannataro e Comito (2003). Esta ontologia mapeia e descreve os principais conceitos e relacionamentos entre os aspectos-chave do domínio de mineração de textos. No processo de construção / adaptação desta ontologia, seguimos os passos propostos por Noy e McGuiness (2001) e as recomendações de Guarino (1998).

Observamos também, que, como sub-produto desta ontologia, foi proposto um processo de validação da mesma, composto da aplicação do método *OntoClean* proposto por Guarino e Welty (2002) combinado com a aplicação de entrevistas sistematizadas com especialistas que validaram não somente a estrutura da ontologia mas também a representação dos conceitos nela contidos.

O *MiningFlow* é baseado em serviços *web* e foi construído para ser independente de qualquer SGWf, o que pode ser considerado um diferencial das ferramentas existentes. Em nossos experimentos pudemos avaliar essa independência ao utilizarmos os SGWf Taverna e Kepler no estudos de observação utilizado para a validação da proposta.

Estão disponibilizados no *MiningFlow* recursos semânticos que ajudam o analista a definir seu experimento de mineração de textos, tanto em alto-nível (baseada nos conceitos que estão definidos e mapeados na ontologia proposta, ou seja, na definição abstrata do *workflow*) quanto em mais baixo nível (em termos de programas ou serviços executáveis, ou seja, a definição concreta do *workflow*) com o apoio da *MF-Ontology* durante o processo de definição dos *workflows*.

Além disso, os dados de proveniência são sempre associados aos conceitos da ontologia, fazendo com que as informações de origem e de execução dos *workflows* tenham semântica conhecida. Como um recurso adicional, porém não menos importante, o *MiningFlow* provê meios para que o analista recupere informações de execuções de experimentos anteriores e possa realizar análises posteriores que forneçam subsídios necessários para a definição de novos experimentos.

Com o protótipo da ferramenta e a ontologia implementados, ambos puderam ser finalmente submetidos a um estudo de observação, seguido por um estudo de caso. Para o estudo de observação da ferramenta, que visava identificar problemas de implementação, além de obviamente executar uma validação em primeira instância do protótipo, contamos com a participação de alunos de mestrado do Programa de Engenharia de Sistemas e Computação (PESC) da COPPE-UFRJ. Os serviços disponibilizados no MiningFlow para esta validação foram adaptados de programas da ferramenta WEKA com o objetivo de representar um ambiente real para analistas de mineração de textos. Já o estudo de caso da ferramenta foi realizado com alunos de

mestrado e doutorado do PESC e do Programa de Engenharia Civil (PEC) da COPPE-UFRJ. Os serviços disponibilizados no MiningFlow para esta validação foram adaptados de programas disponibilizados por alunos do PEC especializados no domínio de mineração de textos.

As análises obtidas no estudo de caso apresentaram evidências quanto à efetividade dos recursos disponíveis aos analistas de mineração de textos no apoio às atividades de definição, execução e análise dos resultados dos experimentos em mineração de textos via *MiningFlow*.

De acordo com as análises realizadas por usuários especialistas e não-especialistas, a separação do experimento em termos de conceitos e programas executáveis foi uma grande vantagem, já que além de incluir semântica no processo, fornece subsídios para que o sistema auxilie na elaboração do experimento no caso dos usuários menos experientes, evitando-se assim que erros simples sejam propagados. Na análise das funcionalidades que foi realizada, todos os entrevistados apontaram o módulo de definição abstrata ou o módulo de definição concreta como um dos dois mais relevantes do sistema.

Outra vantagem reconhecida pelos usuários avaliadores foi o módulo de redefinição dos *workflows* que possibilita ao analista variar os parâmetros de seus experimentos mantendo a unidade semântica do experimento. Como já foi apontado anteriormente neste trabalho, esta variação controlada dos parâmetros é de suma importância em qualquer experimento de mineração de textos.

Além disto, foi apontada como uma grande vantagem a independência do sistema em relação ao SGWf e conseqüentemente à máquina de *workflow* que está sendo utilizada. Desta forma, as definições são sempre armazenadas no repositório da ferramenta, em um esquema relacional, sem estar preso à linguagens de definição proprietárias ou sem padrão definido.

É importante ressaltar que a proposta desta dissertação é apenas uma instância do que pode ser realizado com o potencial das idéias propostas, sem que o protótipo implementado tenha como objetivo cobrir todos os aspectos necessários para a instrumentação de um experimento de mineração de textos. Desta forma, desenvolvemos um protótipo que conseguisse disponibilizar condições básicas de apoio

ao ciclo de vida de um experimento desta natureza, a fim de validar a idéia inicial e se tornar disponível para analistas de mineração de textos.

A partir destas análises, e da avaliação final do protótipo realizado pelos participantes, pudemos concluir que, mesmo que com algumas limitações, o *MiningFlow* cumpre os seus objetivos traçados inicialmente, fornecendo ao analista os recursos necessários para apoiar o ciclo de vida de um experimento de mineração de textos de forma efetiva.

Entretanto, durante o estudo de observação e o estudo de caso (mas, principalmente durante o estudo de caso) recebemos inúmeras sugestões de melhorias na ferramenta que poderiam ser incorporadas ao MiningFlow de forma a criar um ambiente mais amigável e mais completo para o analista de mineração de textos.

Uma das maiores reivindicações foi a falta de uma interface visual e mais amigável para o analista de mineração de textos. Todo o sistema, apesar de ter sido desenvolvido para ser o mais simples possível, ainda é muito textual, o que pode dificultar o trabalho. Entretanto, a construção de MiningFlow nessa dissertação teve o objetivo de prova de conceito. Assim, recursos visuais e gráficos ficaram fora do escopo da dissertação. Uma das melhorias principais a serem aplicadas na implementação é a mudança de interface, fazendo com que a mesma se torne mais intuitiva para o usuário.

Outra requisição dos usuários foi a implantação de um módulo que controlasse o agendamento automático de *workflows* concretos (redefinidos ou não) e que gerenciasse o estado destes experimentos (se já foram executados ou não). O sistema não foi concebido inicialmente para executar um experimento automaticamente, porém é uma requisição plausível e que deve ser levada em consideração ao se realizarem melhorias no mesmo.

Outra melhoria que deve ser desenvolvida é a expansão do módulo de tradução para lidar com mais linguagens de definição de *workflows* científicos. Atualmente para os estudos de observação e de caso, o sistema estava gerando apenas saídas para a linguagem do Taverna e Kepler. Outras modificações, a fim de que o sistema gere saídas em outras linguagens são interessantes. Porém devido ao grande numero e diversidade de SGWf disponíveis, MiningFlow foi construído de modo a tornar simples

a inclusão de novos mapeamentos. O uso do XML facilita essa tarefa e o fato de haver um *workflow* genérico em MT também. Um guia de orientação para essa geração de especificações para novos SGWf encontra-se no portal da MiningFlow.

Outra melhoria a ser desenvolvida é fornecer a possibilidade de o usuário poder cadastrar itens na ontologia (ou mesmo criar novas classes) via o próprio portal *MiningFlow*. Atualmente existe uma complexidade adicional no processo, já que o usuário tem que utilizar uma ferramenta de ontologias (Protégé (2007), por exemplo) para manipular o arquivo de definição da ontologia para, só então, cadastrar as informações no portal.

A partir da base de dados do *MiningFlow*, o analista sênior de mineração de textos também estará apto a aplicar algoritmos de mineração na própria base de definições de experimentos e de resultados intermediários. Através da aplicação destes algoritmos, poderão se extrair padrões de uso de processos de mineração de textos completos.

# Referências Bibliográficas


AALST, W. M. P., HOFSTEDE, A. H.M, BARROS, A. P, 2000, "*Workflow* Patterns". BETA Working Paper Series, WP 47, Eindhoven University of Technology, Eindhoven, 2000.

AALST, W., HEE, K. *Workflow Management: Models, Methods, and Systems*.MIT Press, January 2002.

ALTINTAS, I., et al. "Kepler: An Extensible System for Design and Execution of Scientific *Workflows*", Proceeding of 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04), 21-23 June 2004, Santorini Island, Greece.

ALTINTAS, I., BARNEY, O., JAEGER-FRANK, E., 2006, "Provenance Collection Support in the Kepler Scientific *Workflow* System". IPAW2006, Chicago, Illinois, May 2006.

APACHE, 2007, "Jakarta Commons HttpClient", Apache *Software* Foundation, URL: http://jakarta.apache.org/commons/httpclient/, visitado em 03/06/2007.

APPELT, D. E; ISRAEL, D. J. Introduction to Information Extraction Technology. In Proceedings of the 16th International Joint Conference on Artificial Intelligence, 1999.

APPLE Inc., 2007, "Apple – Mac OS X – Leopard Sneak Peek", URL: http://www.apple.com/macosx/leopard/, visitado em 05/04/2007.

ARKIN, A.,2001, "Business Process Modeling Language (BPML)". Disponível em http://www.bpmi.org/bpml-spec.esp, Março 2001.

AXIS, 2006, "*Web* Services – Axis", Apache *Software* Foundation, URL: http://ws.apache.org/axis/, visitado em 05/04/2007.

BECHHOFER, S. et al. "OilEd: a Reason-able Ontology Editor for the Semantic *Web*". (2001). Available from Internet: <http://potato.cs.man.ac.uk/papers/ki2001.pdf>. Access: 05 Jan. 2002.

BERRY, M. W. *Survey of text mining: clustering, classification, and retrieval.* Springer-Verlag: New York, 2004.

BORKO, H.; BERNICK, M. "Automatic document classification". In Journal of the Association for Computing Machinery, 10:151--162, 1963.

BUNEMAN, P, KHANA, S. and TAN, W. C. 2001, "Why and where: A characterization of data provenance". In Proc. Intl. Conf. on Data Theory (ICDT), volume 1973 of LNCS, pages 316–330. Springer, 2001.

CAI, D., HE, X. and HAN, J. "Document Clustering using locality preserving indexing". In IEEE Trans. Knowledge and Data Engineering, 1624-1637, 2005.

CANNATARO, M., COMITO, C. 2003, "A Data Mining Ontology for Grid Programming", in Workshop on Semantics in Peer-to-Peer and Grid Computing (in conj. with WWW2003), march 2003.

CANNATARO, M., TALIA, D. 2003, "KNOWLEDGE GRID: an architecture for distributed knowledge discovery", Communications of ACM, CACM, Vol. 46, N.1, pp. 89-96, january 2003.

CANNATARO, M., VELTRI, P. 2006, "MS-Analyzer: Composing and Executing Preprocessing and Data Mining Services for Proteomics Applications", in Concurrency and Computation: practice and experience, volume 19, issue 15, pages 2047-2066, december 2006.

CARROLL, S., PAVLOVIC, V. "Protein classification using probabilistic chain graphs and the Gene Ontology structure". Bioinformatics 22(15): 1871-1878 (2006).

CAVALCANTI, M.C.R, CAMPOS, M.L.M, MATTOSO, M.L.Q 2003, *Scientific Resources Management: Towards an In Silico Laboratory.* Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro - Brasil.

CAVALCANTI, M. "Scientific Resources Management: Towards an In Silico Laboratory", Ph.D. Thesis, Technical Report ES-605/03, COPPE/UFRJ, Brazil, 2003.

CERAMI, E., 2002, *Web Services Essentials*, O`Reilly Media Group, 2002.

CHIDANAND, A.; DAMERAU, F.;WEISS, S.M. "Automated Learning of Decision Rules for Text Categorization". ACM Transaction on Information Systems, Vol.12, No.3, July 1994.

CLEMENTINE, 2007, "Text Mining for Clementine", Disponível em: http://www.spss.com.br/tecnologias/text_mining.htm. Acessado em 28 de Dezembro de 2007.

CURBERA, F., GOLAND, Y., ANDREWS, T., et. al, "Business Process Execution Language for *Web* Services v1.1". Microsoft, BEA, IBM, May-2003. Disponível em: http://www.ibm.com/developerworks/library/ws-bpel/.

DOMINGUE, J.; MOTTA, E.; CORCHO, O., 1999, "Knowledge modeling in *Web* Onto and OCML"; a user guide (1999). Available from Internet: <http://kmi.open.ac.uk/projects/*web*onto/user_guide.2.4.pdf> Acesso: 15/12/2007.

FELDMAN, R; DAGAN, I., 1995, "Knowledge discovery in textual databases (KDT)". In proceedings of The First International Conference on Knowledge Discovery and Data Mining (KDD-95), Montreal Canada, August 20-21, AAAI Press, 112-117.

FELDMAN, R., SANGER, J., 2006, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press, 2006.

FELICISSIMO, C.H., SILVA, L.F, BREITMAN, K.K. 2003, "Geração de Ontologias Subsidiada pela Engenharia de Requisitos", in Workshop de Engenharia de Requisitos WER 2003, Piracicaba, Brazil.

FENSEL, D. et al., 2000, "OIL in a nutshell In: Knowledge Acquisition, Modeling, and Management", Proceedings of the European Knowledge Acquisition Conference (EKAW-2000), R. Dieng et al. (eds.), Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag, October.

FREIRE, J. et al., 2006, "VisTrails: visualization meets data management". SIGMOD Conference 2006: pp 745-747.

GLOBUS, 2007, "The Globus Alliance", Disponível em: http://www.globus.org/. Acessado em 27 de Dezembro de 2007.

GOBLE, C., WROE, C. , STEVENS, R., 2003, "The myGrid Project: Services, Architecture and Demonstrator", Proceedings UK e-Science All Hands Meeting 2003 Editors - Simon J Cox, p. 595-603, 2003.

GOBLE, C., WROE, C. , STEVENS, R., 2003, "The myGrid Project: Services, Architecture and Demonstrator", Proceedings UK e-Science All Hands Meeting 2003 Editors - Simon J Cox, p. 595-603, 2003.

GOBLE, C., WROE, C., STEVENS, R., "The myGrid Project: Services, Architecture and Demonstrator", Proceedings UK e-Science All Hands Meeting 2003 Editors - Simon J Cox, p. 595-603, 2003.

GOI, 2007, "Global *Internet* Statistics (by language)". Disponível em: http://www.glreach.com/globstats/index.php3. Acessado em 20 de Dezembro de 2007.

GRAHAM, S., DAVIS, D., SIMEONOV, S., *et al.*, 2005, *Building Web Services with Java – Making sense of XML, SOAP, WSDL e UDDI*, Sams Publishing.

GRUBER, T.R., 1993, *A translation approach to portable ontology specifications – Knowledge Acquisition* – 5: 199-220

GUARINO, N., 1998, "Formal Ontology in Information Systems", Proceeding of FOIS'98, Trento, Italy, IOS Press, June 1998.

GUARINO, N., WELTY, C.,2000. "A formal ontology of properties. In R. Dieng and O. Corby, editors, Knowledge Engineering and Knowledge Management: Methods, Models and Tools". 12th International Conference, EKAW2000, pages 97–112. Springer-Verlag, 2000.

GUARINO, N., WELTY, C., 2002 *"Evaluating Ontological Decisions with OntoClean"*, Communications of the ACM. 2(45):61–65, 2002.

GUIZZARDI, G., 2005 "Ontological Foundations for Structural Conceptual Models", PhD Thesis series, No 05 – 74, Telematica Instituut Fundamental Research Series, Enschede, The Netherlands, 2005.

HAN, J., KAMBER, M. "Data Mining: Concepts and Techniques". San Francisco, USA: Morgan Kaufmann, 2001. pp. 614-628, 364-365, 374

HOFFMAN, T. "Probabilistic latent semantic indexing". In Proc. 1999 Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR`99), 50-57, Berkley, CA, Aug. 1998

HORROCKS, I., PATEL-SCHNEIDER, P. F., VAN HARMELEN, F., 2002, "Reviewing the Design of {DAML+OIL}: An Ontology Language for the Semantic *Web*". Presented at 18th Nat. Conf. on Artificial Intelligence (AAAI2002).

HTTP, 2007, "Hypertext Transfer Protocol", Disponível em: http://www.w3.org/Protocols/, Acessado em 20 de Dezembro de 2007.

INFOMINER, 2007, Disponível em: http://fuzzy.cs.uni-magdeburg.de/InfoMiner/. Acessado em 27 de Dezembro de 2007.

JOACHIMS, T. "A statistical learning model of text classification with support vector machines". In Proc. Int. 2001 ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR`01), 128-136, New Orleans, LA, Sept. 2001.

KEPLER, 2007;Disponível em: http://kepler-project.org/

KOSALA and BLOCKEEL. "*Web* mining research: a survey", SIGKDD Explorations: Newsletter of the Special Interest Group SIG on Knowledge Discovery & Data Mining, 2, 2000.

KPF, 2007, "Kepler Provenance Framework". 2007; Available from: http://kepler-project.org/Wiki.jsp?page=KeplerProvenanceFramework.

LEE, E.A, NEUENDORFFER, A., 2000, "MoML: A Modeling Markup Language in XML Version 0.4". Technical report, University of California at Berkeley, March, 2000.

LEYMANN, F., "*Web* Services Flow Language (WSFL 1.0)", Disponível em http://www-4.ibm.com/*software*/solutions/*web*services/pdf/WSFL.pdf, Maio 2001.

LIBSVM, 2007, "Weka LIBSVM – A Library for support vector machines". Disponível em: http://ntu.csie.org/~cjlin/libsvm/. Acessado em 26 de Dezembro de 2007.

LINUX Online Inc., 2007, "The Linux Home Page at Linux Online", URL: http://www.linux.org/, visitado em 05/04/2007.

LOVINS, J. B. Development of a *stemming* algorithm. Mechanical Translation and Computacional Linguistics, Volume 11, Number 1-2, pages 22-31, 1968.

MAEDCHE, A. et al., 2000, "Representation Language-Neutral Modeling of Ontologies". Avilable from Internet: <http://www.aifb.uni-karlsruhe.de/WBS/Publ/2000/modellierung_amaetal_2000.pdf>. Access: 20 May 2002.

MASTELLA, L.S., ABELI, M., LAMB, L.C, DE ROS, L.F. 2005, "Uma Ontologia Temporal para Modelagem de Conhecimento sobre Ordenação de Eventos", in Encontro Nacional de Inteligência Artificial, São Leopoldo, Brasil.

MATTOSO, M.L.Q. "O que é e-ciência?". 2006. Revista Computação Brasil, SBC, Editorial, p. 2 - 2, dez. 2006.

MEDEIROS, A., PEDRINACI, C., AALST, W., DOMINGUE, J., SONG, M., ROZINAT, A., NORTON, B., and CABRAL, L. (2007) "An Outlook on Semantic Business Process Mining and Monitoring", Workshop: 3rd International IFIP Workshop On Semantic *Web* & *Web* Semantics (SWWS '07) at On The Move Federated Conferences and Workshops.

MEYER, L. "Parallel Strategies for Processing Scientific *Workflows*". COPPE/UFRJ, Brazil, April 2004.

MICROSOFT Corporation, 2007, "Windows Home Page", URL: http://www.microsoft.com/windows/default.mspx, visitado em 05/04/2007.

MIERSWA, I., et al., 2006 "YALE: Rapid Prototyping for Complex Data Mining Tasks". The 12th Annual SIGKDD International Conference on Knowledge Discovery and Data Mining. Philadelphia, USA, 2006.

MIERSWA, I., WURST, M., KLINKENBERG, R. and SCHOLZ, M. "YALE: rapid prototype for complex data mining tasks", Proceedings of the 12th ACM SIGKDD International conference on knowledge discovery and Data Mining, Philadelphia, USA. 2006. pp. 935-940.

MOML, 2007, "Modeling Markup Language in XML", Disponível em: http://www.gigascale.org/pubs/16.html. Acessado em 28 de Dezembro de 2007.

MOREAU, L., LORD, P., WROE, C., STEVENS, R., GOBLE, C., MILES, S., DECKER, K., PAYNE, T. e PAPAY, J., "Semantic and personalised service discovery", Procedings of Workshop on Knowledge Grid and Grid Intelligence (KGGI'03), em conjunto com IEEE/WIC International Conference on *Web* Intelligence/Intelligent Agent Technology, Halifax, Canada, 2003.

MOREIRA, O. V; HUYCK, C.R. *Stemming* Algorithm for the Portuguese Language. In Proceedings of the SPIRE conference, Laguna de San Raphael, Chile, November 13-15, 2001.

MOZILLA Foundation, 2007, "Firefox – Rediscover the *web*", URL: http://www.mozilla.com/en-US/firefox/, visitado em 05/04/2007.

MYGRID. 2008, Disponível em http://www.mygrid.org.uk, visitado em 25 de fevereiro de 2008.

NIGAM, K., MCCALLUM, A., THRUN, S. and MITCHEL, T. "Text classification from labeled and unlabeled documents using EM". Machine Learning, 103-134, 2000.

NOY, N. F., MCGUINNESS, D. L., 2001, "Ontology Development 101: A Guide to Creating Your First Ontology". Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001- 0880, March 2001.

OBO, 2008, "Open Biomedical Ontologies", URL: http://www.obofoundry.org/ontologies.shtml, visitado em 22/02/2008.

PEREIRA, V.B, 2007, *Olimpo: Recomendação de Conhecimento Pessoal Através de Ontologias.* Engenharia de Sistemas e Computação, COPPE/UFRJ, Rio de Janeiro - Brasil.

PMML, 2007, "Predictive Model Markup Language", Disponível em: http://www.dmg.org/pmml-v3-0.html. Acessado em 27 de Dezembro de 2007.

PORTER, M. An algorithm for suffixing stripping. Program, Volume 14, Number 3, pages 130-137, 1980.

PROTEGE, 2007, "The Protégé Ontology Editor and Knowledge Acquisition System", URL: http://protege.stanford.edu/, visitado em 15/12/2007.

SCUFL, 2007, "Simple Conceptual Unified Flow Language", Disponível em: http://www.mygrid.org.uk/usermanual1.7/scufl_language_wb_features.html. Acessado em 27 de Dezembro de 2007.

SEBASTIANI, F. "Machine learning in automated text categorization". ACM computing surveys, 34: 1-47, 2002.

SINGH, M. VOUK, M.,1996, "Scientific *Workflows*: Scientific Computing meets Transactional *Workflows*". In Proceedings of the NSF Workshop on *Workflow* and Process Automation in Information Systems: State-of-the-Art and Future Directions, pp. 28-34, Univ. Georgia, Athens, GA, USA, 1996.

SMTP, 2007, "Simple Mail Transfer Protocol", Disponível em: http://www.ietf.org/rfc/rfc2821.txt/. Acessado em 20 de Dezembro de 2007.

SULLIVAN, D., 2001, *Document Warehousing and Text Mining.* John Wiley & Sons, New York, 2001.

SUN, 2007, Java Technology Disponível em http://java.sun.com/, 2007.

TAN, A., 1999, "Text mining: the state of the art and the challenges". In: Pacific-Asia Workshop on Knowledge Discovery from Advanced Databases-PAKDD'99, Beijing, April (1999), 65-70.

TARGINO, R., 2004, "O Ambiente 10+C para definição e execução de *workflows in silico* através de serviços *web*", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

THATTE, S., "XLANG: *Web* services for Business Process Design". Disponível em http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm, Microsoft Corporation, 2001.

TICHY, W.F., 2000, *Hints for Reviewing Empirical Work in Software Engineering, Empirical Software Engineering*. 5(4): 309-312.

TMRG. "The Text Mining Research Group". 2008. University of Waikato. Disponível em: http://www.cs.waikato.ac.nz/~nzdl/textmining/. Acessado em 22 de Fevereiro de 2008.

TOMCAT, 2007, Apache Tomcat. Disponível em http://jakarta.apache.org/, 2002.

TRAVASSOS, G.H, WERNER, C.M.L, BARROS, M, 2005, "Um Estudo Experimental sobre a Utilização de Modelagem e Simulação no Apoio à Gerência de Projetos de *Software*". In: *XIX Simpósio Brasileiro de Engenharia de Software (SBES)*, Uberlândia, MG, Brasil, Outubro de 2005.

TRIANA. "Open Source Problem Solving Environment". 2007; Available from: http://www.trianacode.org/index.html.

TUM, 2007, "Taverna User Manual", Version 1.5. Disponível em: http://www.umanitoba.ca/afs/plant_science/psgendb/doc/taverna/manual.pdf

UDDI, 2007, "The UDDI Version 3.0.1 Specification", Disponível em: http://www.uddi.org/specification.html, 2002.

VAISLEIOS, H.; GRAVANO, L.; MAGANTI, A., "An Investigation of Linguistic Features and Clusters Algorithms for Topical Document *Clustering*". Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

VAN SOLINGEN, R., BERGHOUT, E., 1999, "The Goal / Question / Metric Method: A Practical Guide for Quality Improvement of *Software* Development", McGrawn Hill. ISBN 0077095537

VARELLA, A., 2007, "COOPRACTICE – Comunidades de prática virtuais apoiadas por ontologias", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

VIRDELL, M., 2003, "Business processes and *workflow* in the *Web* services world". IBM developerWorks, 1 January 2003. Disponível em http://www.106.ibm.com/developerworks/*web*services/library/ws-work.html.

W3C, 1999, "HyperText Markup Language (HTML) Home Page", World Wide *Web* Consortium, URL: http://www.w3.org/TR/html401/, visitado em 12/06/2007.

W3C, 2001, "URIs, URLs, and URNs: Clarifications and Recommendations 1.0", Report from the joint W3C/IETF URI Planning Interest Group, URL: http://www.w3.org/TR/uri-clarification/.

W3C, 2004, "RDF Vocabulary Description Language 1.0: RDF Schema", World Wide *Web* Consortium, URL: http://www.w3.org/TR/rdf-schema/, visitado em 15/12/2007.

W3C, 2006, "Extensible Markup Language (XML)", World Wide *Web* Consortium, URL: http://www.w3.org/XML/, visitado em 15/12/2007.

W3C, 2007a, "*Web* Services Activity", World Wide *Web* Consortium, URL: http://www.w3.org/2002/ws/, visitado em 15/12/2007.

W3C, 2007b, "W3C XML Protocol Working Group", World Wide *Web* Consortium, URL: http://www.w3.org/2000/xp/Group/, visitado em 15/12/2007.

W3C, 2007c, "*Web* Services Description Working Group", World Wide *Web* Consortium, URL: http://www.w3.org/2002/ws/desc/, visitado em 15/12/2007.

W3C, 2007d, "OWL Ontology *Web* Language Reference", World Wide *Web* Consortium, URL: http://www.w3.org/TR/owl-ref/, visitado em 15/12/2007.

W3CNOTE, 2007, "*Web* Services Description Language (WSDL) 1.1". Disponível em: http://www.w3.org/TR/wsdl.html, 2001.

W3CNOTE, 2007b, "Simple Object Access Protocol (SOAP) 1.1", Disponível em: http://www.w3.org/TR/SOAP/, 2000. Acessado em 20 de Dezembro de 2007.

W3CNOTE, 2007c, "Note on Simple Object Access Protocol (SOAP) 1.1", Disponível em: http://www.w3.org/TR/SOAP/, 2000.

WANG, K., ZHOU, S. and LIEW, S. C. "Building hierarchical classifiers using class proximity" In Proc.1999 Int. Conf. Very Large Data Bases (VLDB`99), pp. 363-374, Edinburgh, UK, Sept. 1999.

WEKA, 2007, "Data Mining with Open Source Machine Learning in Java", Diponível em: http://www.cs.waikato.ac.nz/ml/weka/. Acessado em 26 de Dezembro de 2007.

WESKE, M., VOSSEN, G., MEDEIROS, C., 1996, "Scientific *Workflow* Management: WASA Architecture and Applications". In Fachbericht Angewandte Mathematik und Informatik, 03/96-I, 1996.

WFMC, 2007, "*Workflow* Management Coalition", Disponível em: http://www.wfmc.org/. Acessado em 27 de Dezembro de 2007.

YANG, Y.;PEDERSEN, J.P. A Comparative Study on Feature Selection in Text Categorization. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), pp. 412-420, 1997.

YU, J, BUYYA, R. 2005, "A taxonomy of scientific *workflows* systems for grid computing". In ACM SIGMOD Record Archive, volume 34, Issue 3 pages 44–49.2005.

ZHAO, J., GOBLE, C., STEVENS, R., 2004, "Semantic *web* applications to e-science *in silico* experiments". Alternate Track Papers & Posters of WWW 2004: 284-285.

# Questionário de Qualificação do Participante

1) Dados Pessoais

   - Nome: _____
   - Data de nascimento: __/__/____
   - Sexo: ____
   - Correio eletrônico: _____

2) Escolaridade

   - Grau de Escolaridade
     ( ) Não-Graduado
     ( ) Mestre
     ( ) Graduado
     ( ) Doutor
     Em andamento: ( ) sim ( ) não
   - Área de Formação: _____

3) Experiência com mineração de textos (algoritmos e implementação)

   - Você já utilizou algum outro sistema de suporte ao processo de mineração de Textos/Dados?
     ( ) Sim
     ( ) Não
     Qual (is):

   - Você já implementou algoritmos ou aplicações de mineração de Textos/Dados?
     ( ) Sim
     ( ) Não
     Qual (is):

   - Você considera que a utilização de um sistema de suporte ao processo de mineração de textos pode facilitar o trabalho do analista e ao mesmo tempo ser um facilitador para o desenvolvimento de novas tecnologias?
     ( ) Sim
     ( ) Não

   - Você atua em alguma função onde mineração de textos é uma atividade constante?
     ( ) Sim
     ( ) Não

- Qual ?

- Como você se considera sua experiência em mineração de textos?
( )pouco experiente ( )experiência razoável ( )experiente ( )muito experiente

- Você já leu alguma publicação especifica sobre o assunto (livros, artigos, relatórios técnicos, etc)?
  ( ) sim
  ( ) não
- Se sim, poderia citar o nome de um dos autores ou da publicação?

4) Experiência com *workflows* científicos

- Você já utilizou algum sistema gerenciador de *workflows* científicos?
  ( ) sim
  ( ) não

Qual (is):

- Qual a sua experiência com SGWf?
( )pouco experiente ( )experiência razoável ( )experiente ( )muito experiente

- Você atua (ou já atuou) em alguma função onde a utilização de *workflows* é uma atividade constante?

  ( ) Sim
  ( ) Não

- Qual ?

- Você já leu alguma publicação especifica sobre o assunto?
  ( ) sim
  ( ) não
- Se sim, poderia citar o nome de um dos autores ou da publicação?

# Questionário de Avaliação Qualitativa do MiningFlow

1) Você acha que as funcionalidades oferecidas pelas tecnologias de *Workflow* podem ser úteis no processo de mineração de textos?

Inútil ○ ○ ○ ○ ○ Extremamente útil

2) Você acha que a modelagem do processo de mineração de textos como uma série encadeada de processos (*Workflow*) pode facilitar o desenvolvimento de novas aplicações e a reuso de implementações?

( ) Sim ( ) Não

3) Você acha que a definição de um *Workflow* abstrato (*templates*) para guiar usuários menos experientes pode ser útil no desenvolvimento de aplicações de mineração de texto, evitando-se assim a propagação de erros durante o processo?

( ) Sim ( ) Não

4) Você acha que a classificação do processo de mineração de texto através de ontologias, bem como a classificação dos *softwares* (ou serviços) pode oferecer uma vantagem no desenvolvimento de uma nova aplicação?

( ) Sim ( ) Não

5) Você acha que as funcionalidades do MiningFlow listadas abaixo podem facilitar o desenvolvimento de uma nova aplicação de mineração de textos?

a. Criação do *Workflow* Abstrato
( ) Nunca ( ) Eventualmente ( )Sempre

b. Criação do *Workflow* Concreto
( ) Nunca ( ) Eventualmente ( )Sempre

c. Armazenamento dos dados de proveniência

( ) Nunca ( ) Eventualmente ( )Sempre

d. Visualização dos Dados de Proveniência
( ) Nunca ( ) Eventualmente ( )Sempre

e. Utilização de uma ontologia para a classificação dos serviços e mapeamento do processo para guiar o usuário durante o desenvolvimento
( ) Nunca ( ) Eventualmente ( )Sempre

f. Checking semântico de serviços (verificar se a entrada de um serviço e a saída do anterior na cadeia são do mesmo tipo)
( ) Nunca ( ) Eventualmente ( )Sempre

6) Você acha que as consultas disponibilizadas para análise dos dados de proveniência são suficientes?
( ) Sim ( ) Não

Se não, quais as consultas sentiu falta?

_____

7) Você acha que ter acesso aos *workflows* gerados por outros usuários pode ser um facilitador na hora de gerar uma nova aplicação?
( ) Sim ( ) Não

8) Saber quais os parâmetros geraram determinadas saídas é uma funcionalidade útil no processo de mineração de textos?
Inútil O O O O O Extremamente útil

9) Qualifique os módulos listados abaixo (quanto à sua implementação):
- Controle de Acesso
Inútil O O O O O Extremamente útil
- Definição do *Workflow* Abstrato
Inútil O O O O O Extremamente útil
- Definição do *Workflow* Concreto
Inútil O O O O O Extremamente útil
- Re-definição do *Workflow* Concreto
Inútil O O O O O Extremamente útil

- Análise dos Dados de Proveniência

Inútil O O O O O Extremamente útil

- Registro dos Dados de Proveniência

Inútil O O O O O Extremamente útil

- Ontologia para Text Mining

Inútil O O O O O Extremamente útil

10) Numere os módulos do MiningFlow a seguir de acordo com o grau de utilidade que você atribui a cada um deles? A qualificação deve ser atribuída em ordem crescente.

_____ - Controle de Acesso

_____ - Definição do *Workflow* Abstrato

_____ - Definição do *Workflow* Concreto

_____ - Re-definição do *Workflow* Concreto

_____ - Análise dos Dados de Proveniência

_____ - Registro dos Dados de Proveniência

_____ - Ontologia para Text Mining

11) Você considera que o MiningFlow atende o objetivo proposto que é dar suporte ao processo de desenvolvimento de aplicações de mineração de texto através da modelagem do processo como um *workflow* científico?

( ) Nunca ( ) Eventualmente ( ) Sempre

12) Você considera uma vantagem o MiningFlow não estar associado diretamente a um SGWf, podendo gerar saídas para qualquer um dos sistemas existentes?

( ) Sim ( ) Não

13) Você utilizaria o MiningFlow como forma de apoiar o desenvolvimento de aplicações de mineração de texto?

( ) Sim ( ) Não

14) Você recomendaria a utilização do MiningFlow como forma de apoiar futuros desenvolvimentos de aplicações de mineração de texto?

( ) Sim ( ) Não

15) O quão fácil ou difícil você achou usar o MiningFlow?

Extremamente difícil O O O O O Extremamente fácil

16) Alguma observação, sugestão, crítica ou comentário a serem feitos?

# Arquivo de definição da *MF-Ontology*

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="Software_Counting">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Software_PreProcessing"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Software Counting</rdfs:label>
 </owl:Class>
 <owl:Class rdf:ID="Software_Stemming">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Software_PreProcessing"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Software Stemming</rdfs:label>
 </owl:Class>
 <owl:Class rdf:ID="Software_Clustering">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Software Clustering</rdfs:label>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Software_Text_Mining"/>
  </rdfs:subClassOf>
 </owl:Class>
```

```
<owl:Class rdf:ID="Algorithm_Remove_Stop_Words">
 <rdfs:subClassOf>
  <owl:Class rdf:ID="Algorithm_PreProcessing"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Method_Based_Neural_Networks">
 <rdfs:subClassOf>
  <owl:Class rdf:ID="Method"/>
 </rdfs:subClassOf>
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Method Based Neural Networks</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Software_Cleaning">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Software Cleaning</rdfs:label>
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Software_PreProcessing"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Function_PreProcessing">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Function PreProcessing</rdfs:label>
 <rdfs:subClassOf>
  <owl:Class rdf:ID="Step_PreProcessing"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Data"/>
<owl:Class rdf:ID="Algorithm"/>
<owl:Class rdf:ID="Method_Based_Trees">
 <rdfs:subClassOf rdf:resource="#Method"/>
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Method Based Trees</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Function_Stemming">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Function Stemming</rdfs:label>
 <rdfs:subClassOf rdf:resource="#Function_PreProcessing"/>
```

```
</owl:Class>
<owl:Class rdf:ID="Software_Remove_Stop_Words">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Software_PreProcessing"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Task_Clustering">
 <rdfs:subClassOf>
  <owl:Class rdf:ID="Task_Text_Mining"/>
 </rdfs:subClassOf>
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Task Clustering</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Function_Plottering">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Function Plottering</rdfs:label>
 <rdfs:subClassOf>
  <owl:Class rdf:ID="Function_Visualization"/>
 </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Stemming">
 <rdfs:subClassOf>
  <owl:Class rdf:about="#Algorithm_PreProcessing"/>
 </rdfs:subClassOf>
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Algorithm Stemming</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Composite_Software">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Composite Software</rdfs:label>
 <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 <rdfs:subClassOf>
  <owl:Restriction>
   <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
   >2</owl:minCardinality>
   <owl:onProperty>
    <owl:FunctionalProperty rdf:ID="IMPLEMENTS_GROUP_OF"/>
```

```
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Step_KDT">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Step KDT</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Task_Classification">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Task_Text_Mining"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Task Classification</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Function_Visualization">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Step_Visualization"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Function Visualization</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Input_Data">
  <rdfs:subClassOf rdf:resource="#Data"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Input Data</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Software_Generate_BOW">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Software_PreProcessing"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Software_Text_Mining">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Software Text Mining</rdfs:label>
  <rdfs:subClassOf>
   <owl:Class rdf:ID="Software"/>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Software_Plottering">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Software Plottering</rdfs:label>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Software_Visualization"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Algorithm_PreProcessing">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithm PreProcessing</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Algorithm"/>
</owl:Class>
<owl:Class rdf:about="#Step_PreProcessing">
  <rdfs:subClassOf rdf:resource="#Step_KDT"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Step PreProcessing</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Function_Generate_BOW">
  <rdfs:subClassOf rdf:resource="#Function_PreProcessing"/>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Visualization">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithm Visualization</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Algorithm"/>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Plottering">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithm Plottering</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Algorithm_Visualization"/>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Classification">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Algorithm_Text_Mining"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
  >Algorithm Classification</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Clustering">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Algorithm_Text_Mining"/>
  </rdfs:subClassOf>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithm Clustering</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Step_Text_Mining">
  <rdfs:subClassOf rdf:resource="#Step_KDT"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Step Text Mining</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Output_Data">
  <rdfs:subClassOf rdf:resource="#Data"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Output Data</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Method_Based_Fuzzy_Logic">
  <rdfs:subClassOf rdf:resource="#Method"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Method Based Fuzzy Logic</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Generate_BOW">
  <rdfs:subClassOf rdf:resource="#Algorithm_PreProcessing"/>
</owl:Class>
<owl:Class rdf:ID="Function_Remove_Stop_Words">
  <rdfs:subClassOf rdf:resource="#Function_PreProcessing"/>
</owl:Class>
<owl:Class rdf:ID="Algorithm_Counting">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Algorithm Counting</rdfs:label>
  <rdfs:subClassOf rdf:resource="#Algorithm_PreProcessing"/>
</owl:Class>
<owl:Class rdf:about="#Step_Visualization">
  <rdfs:subClassOf rdf:resource="#Step_KDT"/>
```

<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Step Visualization</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="*Software*_Classification">
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>*Software* Classification</rdfs:label>
<rdfs:subClassOf rdf:resource="#*Software*_Text_Mining"/>
</owl:Class>
<owl:Class rdf:ID="Function_Counting">
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Function Counting</rdfs:label>
<rdfs:subClassOf rdf:resource="#Function_PreProcessing"/>
</owl:Class>
<owl:Class rdf:about="#*Software*_Visualization">
<rdfs:subClassOf rdf:resource="#*Software*"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>*Software* Visualization</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#*Software*_PreProcessing">
<rdfs:subClassOf rdf:resource="#*Software*"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>*Software* PreProcessing</rdfs:label>
</owl:Class>
<owl:Class rdf:about="#Task_Text_Mining">
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Task Text Mining</rdfs:label>
<rdfs:subClassOf rdf:resource="#Step_Text_Mining"/>
</owl:Class>
<owl:Class rdf:about="#Algorithm_Text_Mining">
<rdfs:subClassOf rdf:resource="#Algorithm"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Algorithm Text Mining</rdfs:label>
</owl:Class>
<owl:Class rdf:ID="Function_Cleaning">
<rdfs:subClassOf rdf:resource="#Function_PreProcessing"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Function Cleaning</rdfs:label>

```
</owl:Class>
<owl:Class rdf:ID="Algorithm_Cleaning">
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Algorithm Cleaning</rdfs:label>
 <rdfs:subClassOf rdf:resource="#Algorithm_PreProcessing"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="CLUSTERS">
 <rdfs:domain rdf:resource="#Composite_Software"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="RESTRICTIONS">
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Data"/>
    <owl:Class rdf:about="#Composite_Software"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:FunctionalProperty rdf:ID="KDT_Ontology_Slot_19">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="TYPE">
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 <rdfs:domain>
  <owl:Class>
   <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Software"/>
    <owl:Class rdf:about="#Composite_Software"/>
   </owl:unionOf>
  </owl:Class>
 </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="INSTITUTION">
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="DERIVED_FROM">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >DERIVED FROM</rdfs:label>
  <rdfs:domain rdf:resource="#Software"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:about="#IMPLEMENTS_GROUP_OF">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >IMPLEMENTS GROUP OF</rdfs:label>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Composite_Software"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="DATA_TYPE">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain rdf:resource="#Data"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >DATA TYPE</rdfs:label>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="PRECEEDS">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Step_KDT"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="WRITES">
  <rdfs:domain>
   <owl:Class>
```

```
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Software"/>
       <owl:Class rdf:about="#Composite_Software"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="NAME">
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Step_KDT"/>
       <owl:Class rdf:about="#Data"/>
       <owl:Class rdf:about="#Algorithm"/>
       <owl:Class rdf:about="#Software"/>
       <owl:Class rdf:about="#Method"/>
       <owl:Class rdf:about="#Composite_Software"/>
      </owl:unionOf>
     </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="COMENTARY">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain>
     <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
       <owl:Class rdf:about="#Step_KDT"/>
       <owl:Class rdf:about="#Data"/>
       <owl:Class rdf:about="#Algorithm"/>
       <owl:Class rdf:about="#Software"/>
       <owl:Class rdf:about="#Method"/>
       <owl:Class rdf:about="#Composite_Software"/>
      </owl:unionOf>
```

```
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="KDT_Ontology_Slot_18">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="IMPROVES_QUALITY">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Step_KDT"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >IMPROVES QUALITY</rdfs:label>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="PATH">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Software"/>
        <owl:Class rdf:about="#Composite_Software"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="READS">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Software"/>
        <owl:Class rdf:about="#Composite_Software"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="BASED_ON">
```

```
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Algorithm"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >BASED ON</rdfs:label>
 </owl:FunctionalProperty>
 <owl:FunctionalProperty rdf:ID="KDT_Ontology_Slot_79">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 </owl:FunctionalProperty>
 <owl:FunctionalProperty rdf:ID="IMPROVES_QUALITY_OF">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >IMPROVES QUALITY OF</rdfs:label>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
 </owl:FunctionalProperty>
 <owl:FunctionalProperty rdf:ID="AUTHOR">
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Algorithm"/>
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Method"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
 </owl:FunctionalProperty>
 <owl:FunctionalProperty rdf:ID="PERFORMS">
```

```
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="IMPLEMENTS_OF">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >IMPLEMENTS OF</rdfs:label>
  <rdfs:domain rdf:resource="#Software"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="KDT_Ontology_Slot_67">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="KDT_Ontology_Slot_17">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="SAME_OF">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <rdfs:domain rdf:resource="#Data"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >SAME OF</rdfs:label>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="VERSION">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Composite_Software"/>
```

```
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="OBJECTIVE">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Step_KDT"/>
     <owl:Class rdf:about="#Data"/>
     <owl:Class rdf:about="#Algorithm"/>
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Method"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="AVAILABLE">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  <rdfs:domain rdf:resource="#Software"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="WSDL_PATH">
  <rdfs:domain>
   <owl:Class>
    <owl:unionOf rdf:parseType="Collection">
     <owl:Class rdf:about="#Software"/>
     <owl:Class rdf:about="#Composite_Software"/>
    </owl:unionOf>
   </owl:Class>
  </rdfs:domain>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
 >WSDL PATH</rdfs:label>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:FunctionalProperty>
<owl:FunctionalProperty rdf:ID="SPECIFIES">
 <rdfs:domain rdf:resource="#Method"/>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
<Task_Clustering rdf:ID="ClusterizaÃ§Ã£o">
 <NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Tarefa de ClusterizaÃ§Ã£o</NAME>
 <OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Clusterizar uma SÃ©rie de Documentos.</OBJECTIVE>
 <COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Tarefa de ClusterizaÃ§Ã£o</COMENTARY>
</Task_Clustering>
<Software_Clustering rdf:ID="EM">
 <COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Serve pra Clusterizar documentos</COMENTARY>
 <WSDL_PATH rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >http://127.0.0.1:8080/axis/services/ClusterizadorEM?wsdl</WSDL_PATH>
 <READS>
  <Input_Data rdf:ID="CaminhodoArquivo">
   <NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >Caminho do Arquivo</NAME>
   <COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >String que ContÃ©m o caminho do arquivo  a ser analisado</COMENTARY>
   <DATA_TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >String</DATA_TYPE>
  </Input_Data>
 </READS>
 <PERFORMS rdf:resource="#ClusterizaÃ§Ã£o"/>
 <NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >Clusterizador EM</NAME>
 <AVAILABLE rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
 >true</AVAILABLE>
 <VERSION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
 >1.0</VERSION>
```

<TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Service</TYPE>
<WRITES rdf:resource="#CaminhodoArquivo"/>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Clusterizar uma ColeÃ§Ã£o de Documentos</OBJECTIVE>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>WEKA</AUTHOR>
<INSTITUTION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Universidade de Waikato</INSTITUTION>
</*Software*_Clustering>
<Function_Remove_Stop_Words rdf:ID="StopWords">
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Remover Stop Words</OBJECTIVE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Stop Words</NAME>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>RemoÃ§Ã£o de Stop Words</COMENTARY>
</Function_Remove_Stop_Words>
<Function_Cleaning rdf:ID="Limpeza">
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>FunÃ§Ã£o de Limpeza</NAME>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Limpar os Dados de Entrada</OBJECTIVE>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Limpa os Dados de Entrada</COMENTARY>
</Function_Cleaning>
<Method_Based_Trees rdf:ID="Method_Based_Trees_7">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Utilza Arvores para Classificar</COMENTARY>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>MÃ©todo Baseado em Trees ADCM</NAME>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Utilizar Arvores para Classificar</OBJECTIVE>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Daniel Cardoso Moraes de Oliveira</AUTHOR>
</Method_Based_Trees>
<*Software*_Stemming rdf:ID="StemmingR">

<READS>
<Input_Data rdf:ID="CaminhodoDiretorio">
<DATA_TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>String</DATA_TYPE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Caminho do DiretÃ³rio</NAME>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>String que ContÃ©m o caminho do diretÃ³rio a ser analisado</COMENTARY>
</Input_Data>
</READS>
<VERSION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>1.0</VERSION>
<AVAILABLE rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>true</AVAILABLE>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Stemmiza</COMENTARY>
<PERFORMS>
<Function_Stemming rdf:ID="Stemming">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Stemmiza o conjunto de palavras</COMENTARY>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Retirar os Radicais das palavras</OBJECTIVE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>FunÃ§ao de StemizaÃ§Ã£o</NAME>
</Function_Stemming>
</PERFORMS>
<WSDL_PATH rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://127.0.0.1:8080/axis/services/runStemmer?wsdl</WSDL_PATH>
<INSTITUTION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>PEC</INSTITUTION>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Stemmizador</NAME>
<WRITES rdf:resource="#CaminhodoDiretorio"/>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Stemmiza</OBJECTIVE>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Renan</AUTHOR>

<TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>ServiÃ§o</TYPE>
</*Software*_Stemming>
<Function_Counting rdf:ID="Contagem">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Conta as Palavras</COMENTARY>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Contar as Palavras</OBJECTIVE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>FunÃ§Ã£o de Contagem</NAME>
</Function_Counting>
<*Software*_Generate_BOW rdf:ID="GerarBOW">
<READS rdf:resource="#CaminhodoDiretorio"/>
<WRITES rdf:resource="#CaminhodoArquivo"/>
<INSTITUTION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>PEC</INSTITUTION>
<AVAILABLE rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>true</AVAILABLE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>GErar BOW</NAME>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Gera BOW</COMENTARY>
<VERSION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>1.0</VERSION>
<WSDL_PATH rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://127.0.0.1:8080/axis/services/generateBOW?wsdl</WSDL_PATH>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Renan</AUTHOR>
<PERFORMS>
<Function_Generate_BOW rdf:ID="BOW">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Gera uma BOW</COMENTARY>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Gerar Bow</OBJECTIVE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>BOW</NAME>
</Function_Generate_BOW>

</PERFORMS>
</*Software*_Generate_BOW>
<*Software*_Cleaning rdf:ID="LimpezaR">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Faz a Limpeza da ColeÃ§Ã£o</COMENTARY>
<WRITES rdf:resource="#CaminhodoDiretorio"/>
<INSTITUTION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>PEC - UFRJ</INSTITUTION>
<READS rdf:resource="#CaminhodoDiretorio"/>
<VERSION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>1.0</VERSION>
<OBJECTIVE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Limpar uma ColeÃ§Ã£o.</OBJECTIVE>
<AVAILABLE rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>true</AVAILABLE>
<WSDL_PATH rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://127.0.0.1:8080/axis/services/loadTextCollection?wsdl</WSDL_PATH>
<TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>ServiÃ§o</TYPE>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>ServiÃ§o de Limpeza</NAME>
<PERFORMS rdf:resource="#Limpeza"/>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Renan</AUTHOR>
</*Software*_Cleaning>
<Input_Data rdf:ID="ArquivoString">
<DATA_TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>String</DATA_TYPE>
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Arquivo de Entrada em String</COMENTARY>
</Input_Data>
<*Software*_Remove_Stop_Words rdf:ID="StopWordsR">
<COMENTARY rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Serve para retirar Stop Words</COMENTARY>
<AUTHOR rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>Renan</AUTHOR>
<PERFORMS rdf:resource="#StopWords"/>

<INSTITUTION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>PEC - UFRJ</INSTITUTION>
<VERSION rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>1.0</VERSION>
<TYPE rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>ServiÃ§o</TYPE>
<READS rdf:resource="#CaminhodoDiretorio"/>
<WRITES rdf:resource="#CaminhodoDiretorio"/>
<AVAILABLE rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>true</AVAILABLE>
<WSDL_PATH rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>http://127.0.0.1:8080/axis/services/removeStopWords?wsdl</WSDL_PATH>
<NAME rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>RemoÃ§Ã£o de Stop Words</NAME>
</*Software*_Remove_Stop_Words>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2, Build 355) http://protege.stanford.edu
-->

# Anexo D

## Gráficos de Avaliação do *MiningFlow*

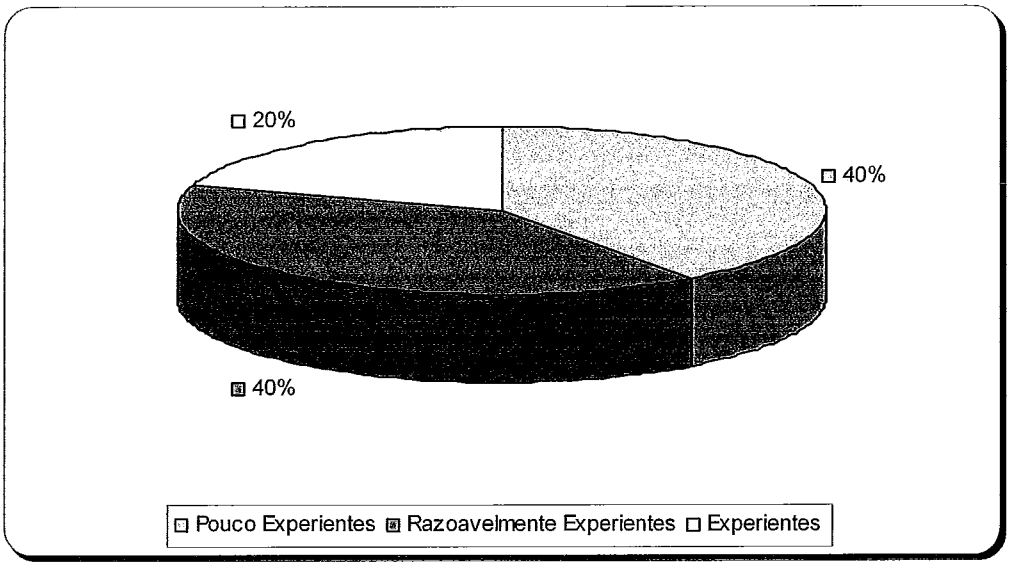


**Figura B.1 – Distribuição dos participantes pelo conhecimento em *workflows* científicos (estudo de observação)**

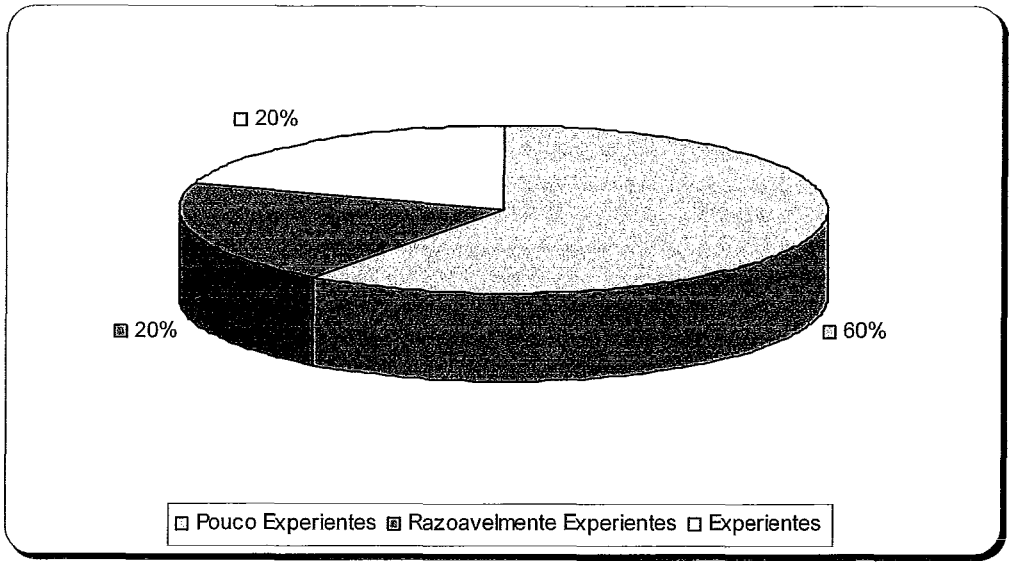


**Figura B.2 – Distribuição dos participantes pelo conhecimento em Mineração de textos (estudo de observação)**

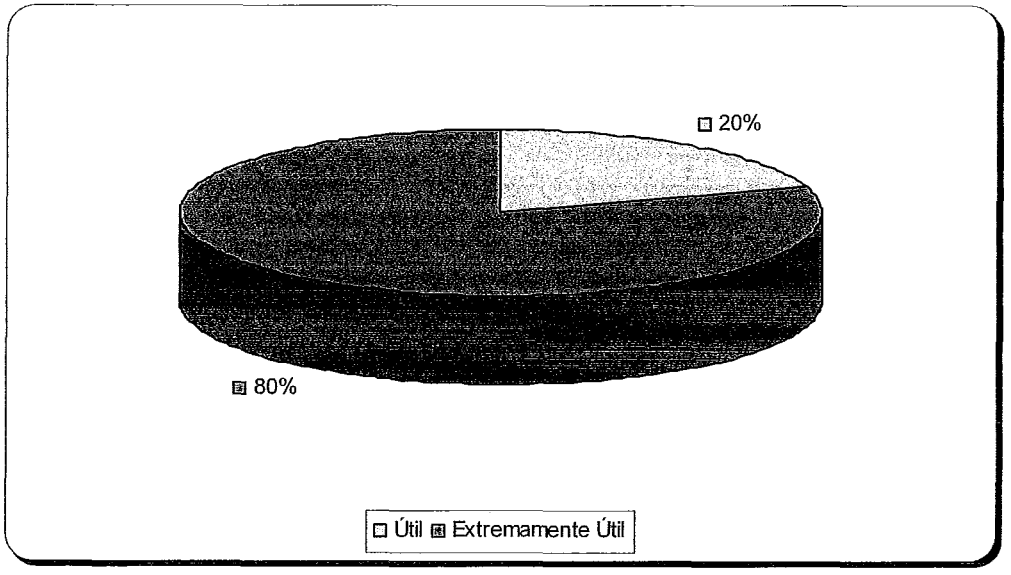


**Figura B.3 – Distribuição das respostas quanto à utilidade da aplicação de técnicas de *workflows* científicos no domínio de mineração de textos (estudo de observação).**

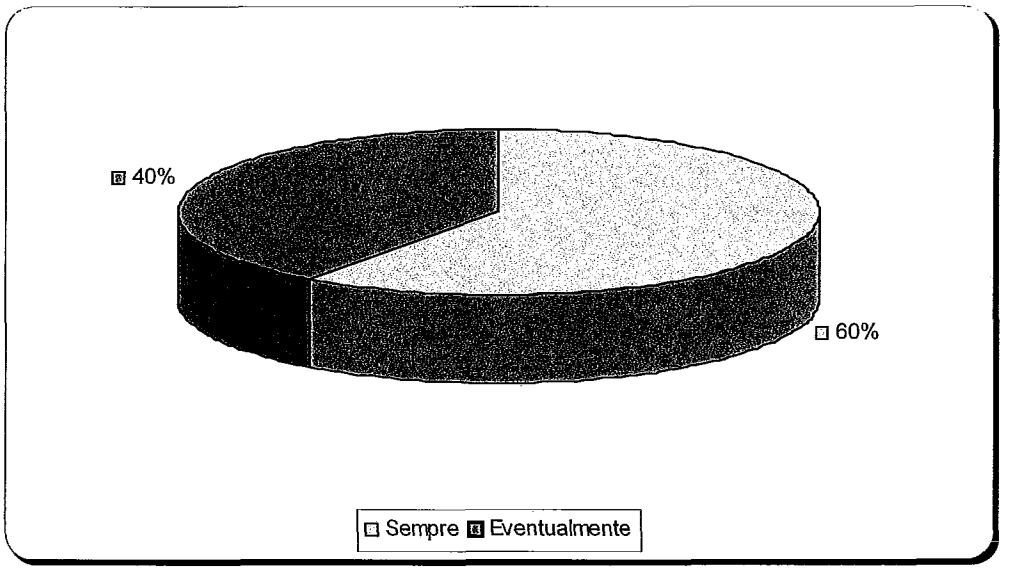


**Figura B.4 – Distribuição das respostas quanto à utilidade do módulo de definição abstrata (estudo de observação).**

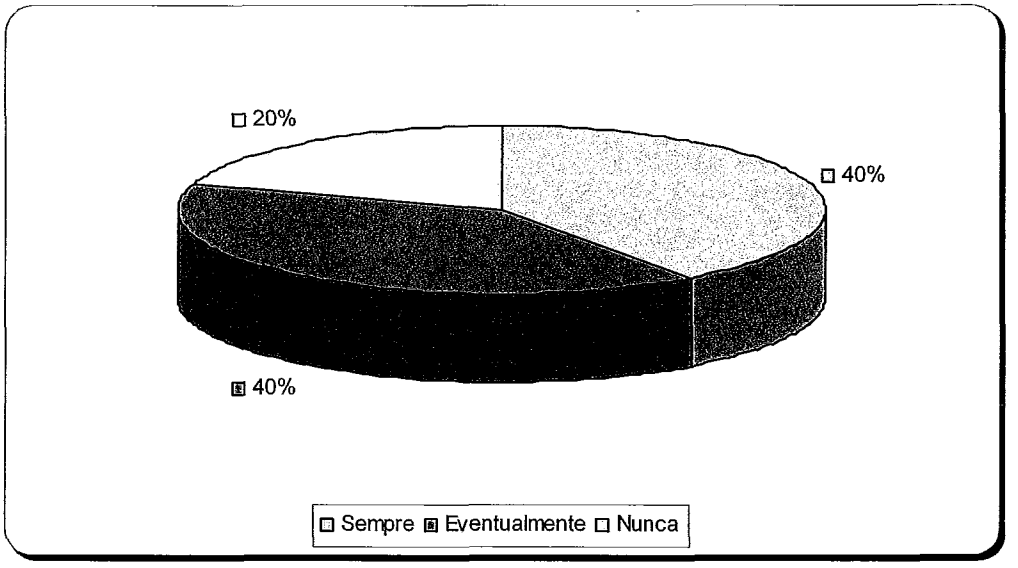


**Figura B.5 – Distribuição das respostas quanto à utilidade do armazenamento de dados de proveniência (estudo de observação)**

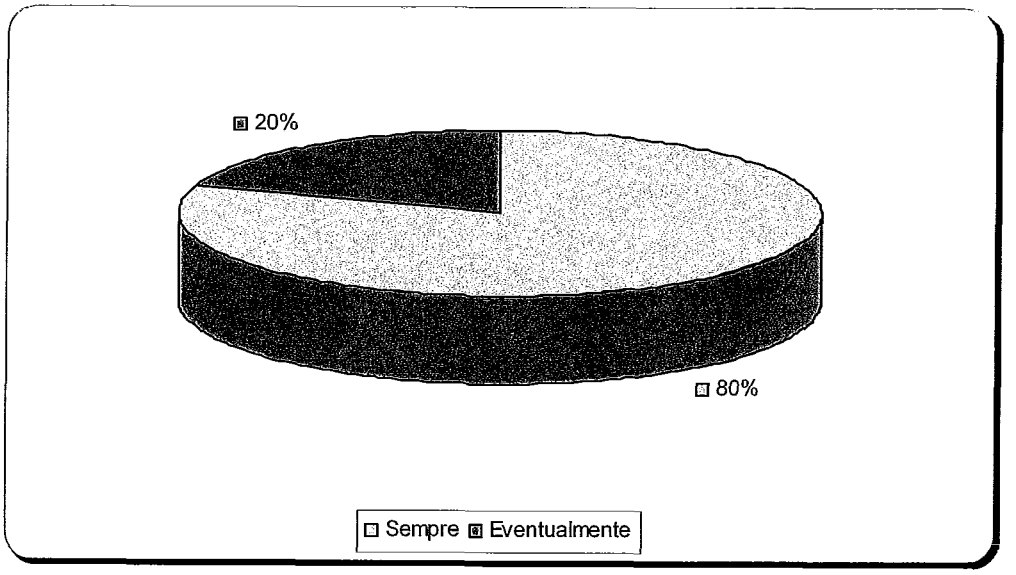


**Figura B.6 – Distribuição das respostas quanto à utilidade da utilização de ontologias na ferramenta (estudo de observação)**

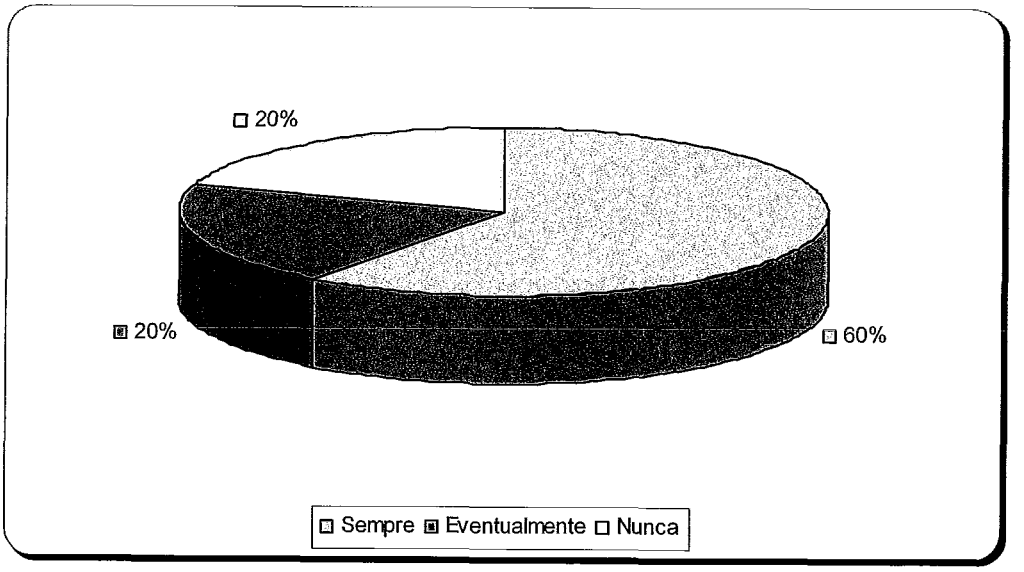


**Figura B.7 – Distribuição das respostas quanto à utilidade do *checking* semântico (estudo de observação)**

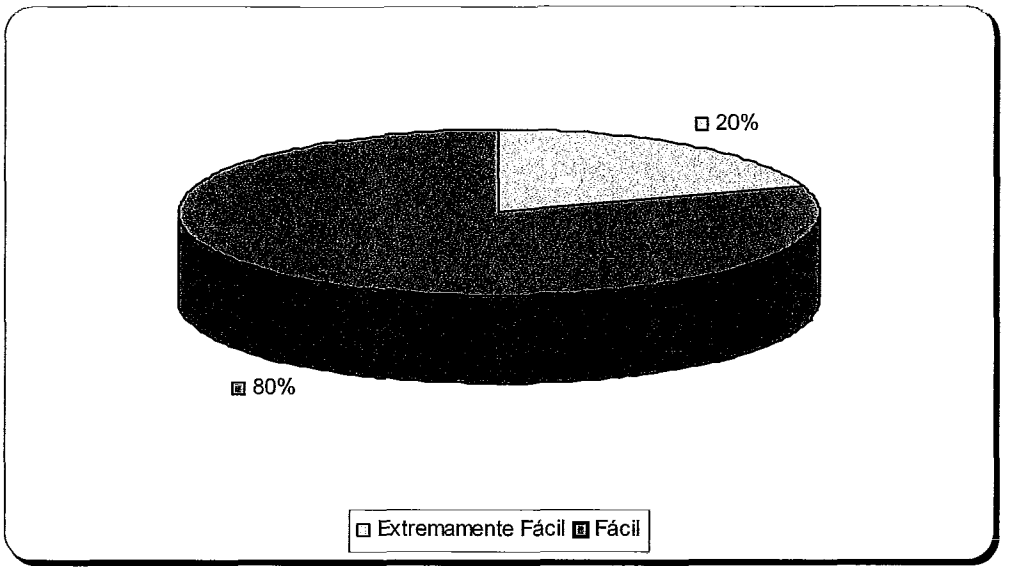


**Figura B.8 – Distribuição das respostas quanto à facilidade de uso (estudo de observação)**

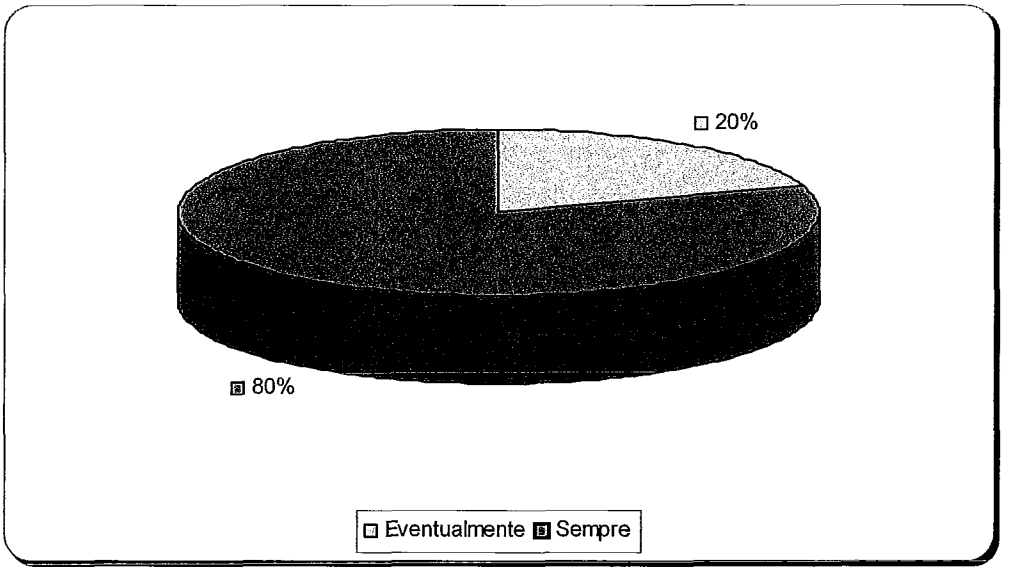


**Figura B.9 – Distribuição das respostas quanto à aderência da proposta aos seus objetivos iniciais (estudo de observação)**

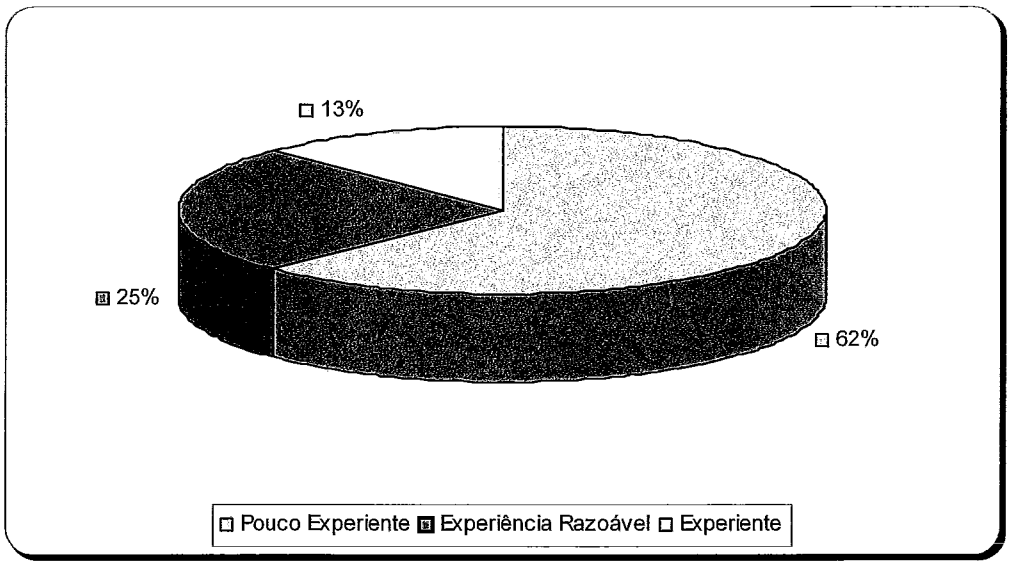


**Figura B.10 – Distribuição dos participantes pelo conhecimento em *workflows* científicos (estudo de caso)**

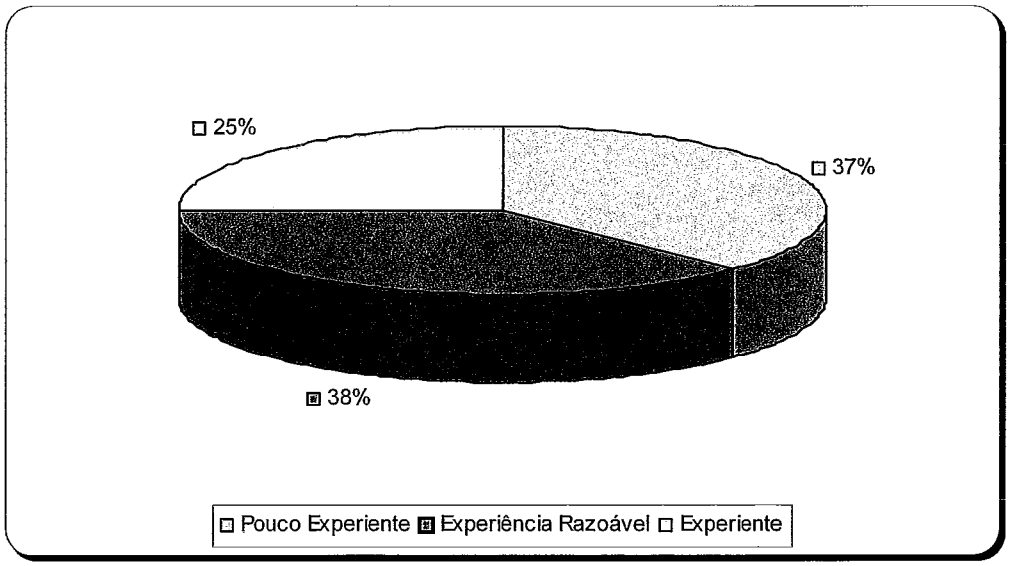


**Figura B.11 – Distribuição dos participantes pelo conhecimento em Mineração de textos (estudo de caso)**

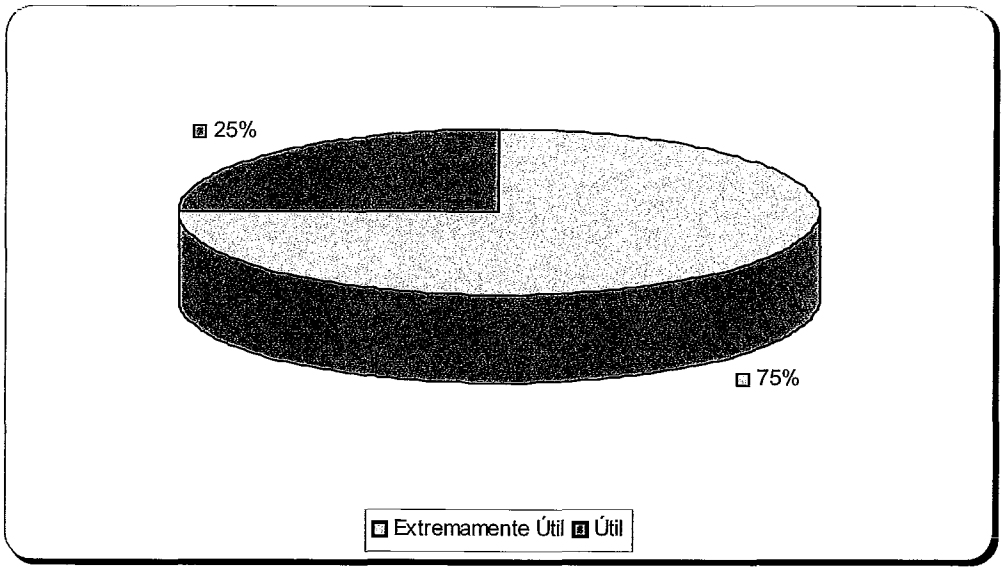


**Figura B.12- Distribuição das respostas quanto à utilidade da aplicação de técnicas de *workflows* científicos no domínio de mineração de textos (estudo de caso)**

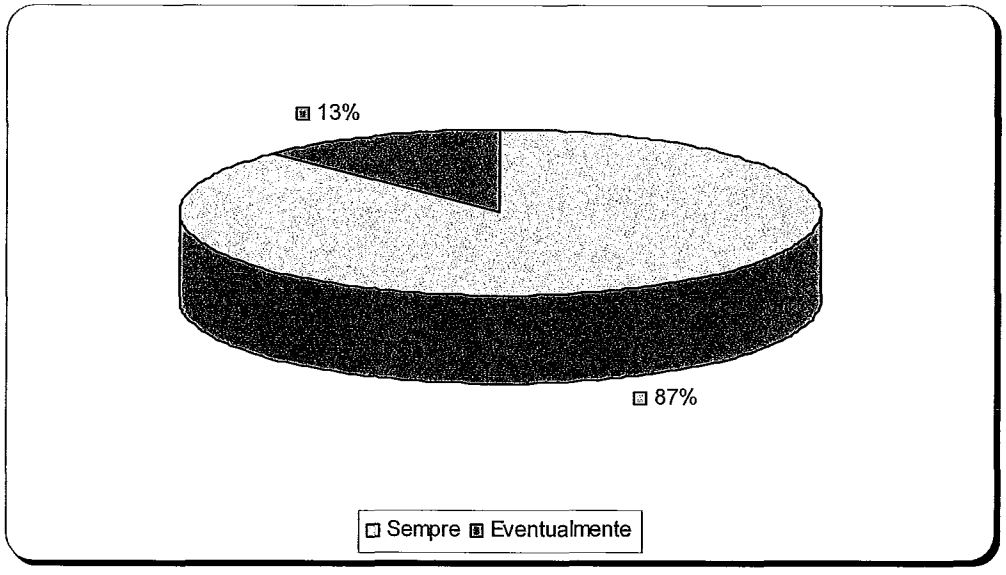


**Figura B.13 – Distribuição das respostas quanto à utilidade do módulo de definição abstrata (estudo de caso)**

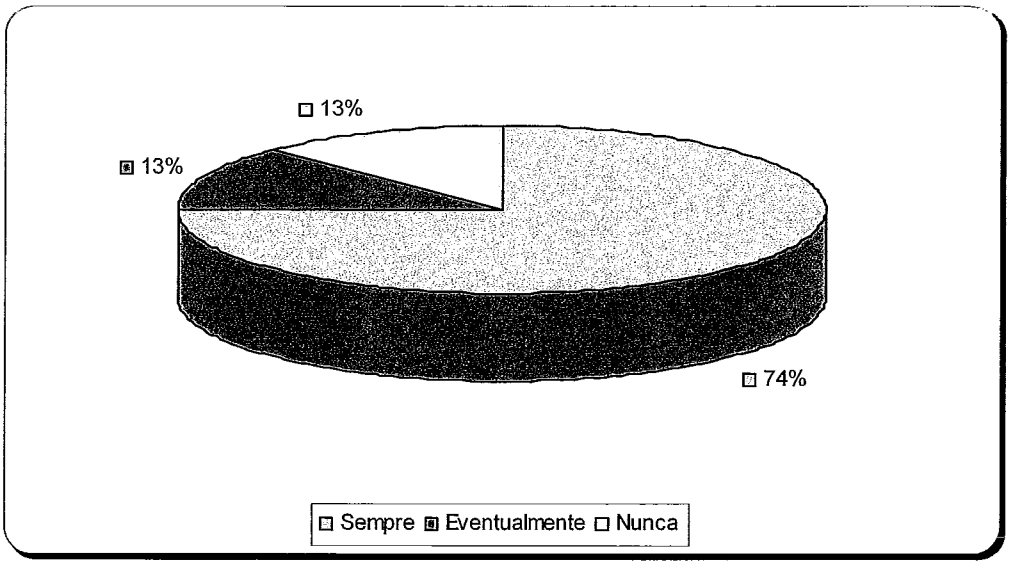


**Figura B.14 – Distribuição das respostas quanto à utilidade do armazenamento de dados de proveniência (estudo de caso)**

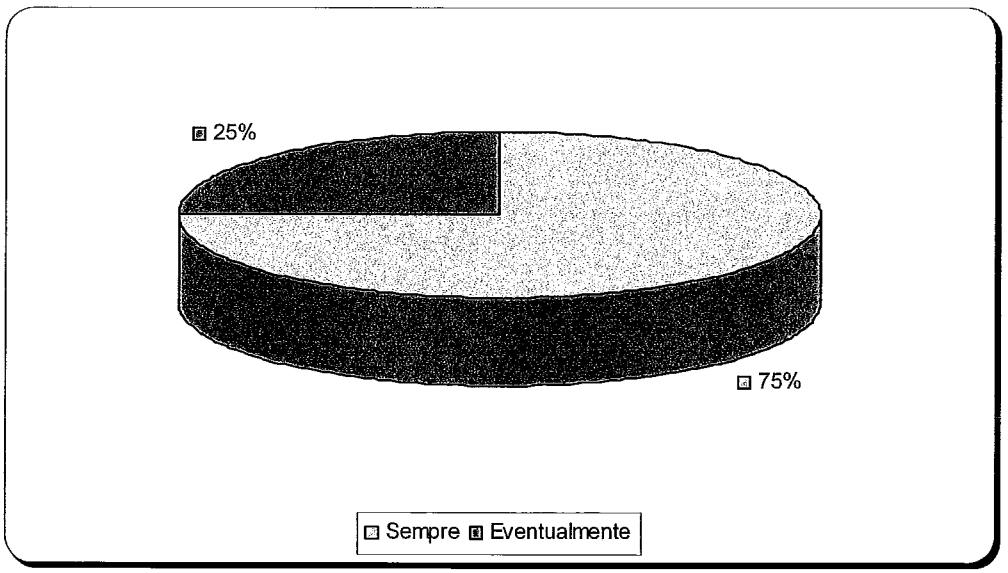


**Figura B.15 – Distribuição das respostas quanto à utilidade da utilização de ontologias na ferramenta (estudo de caso)**

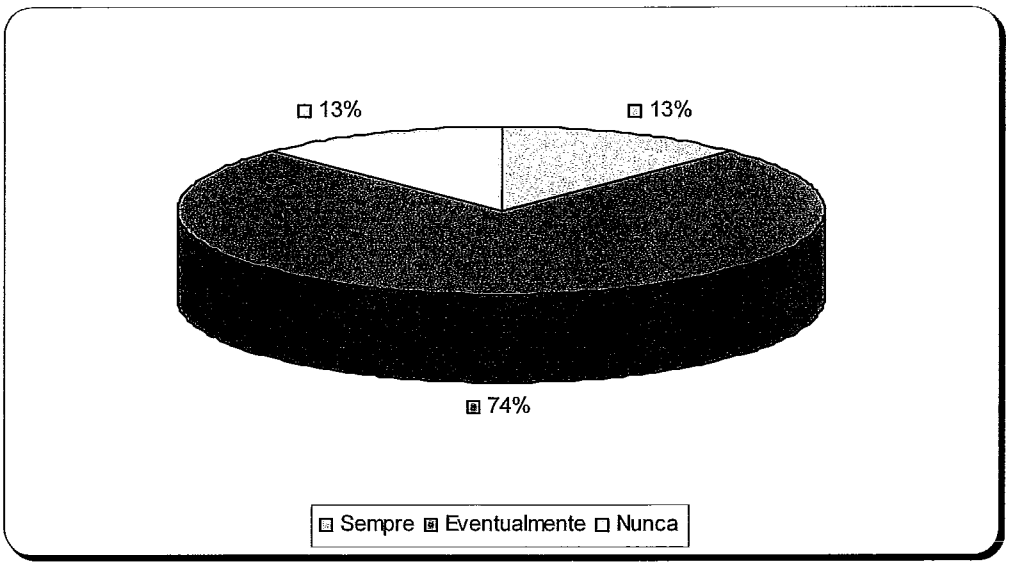


**Figura B.16 – Distribuição das respostas quanto à utilidade do checking semântico (estudo de caso)**

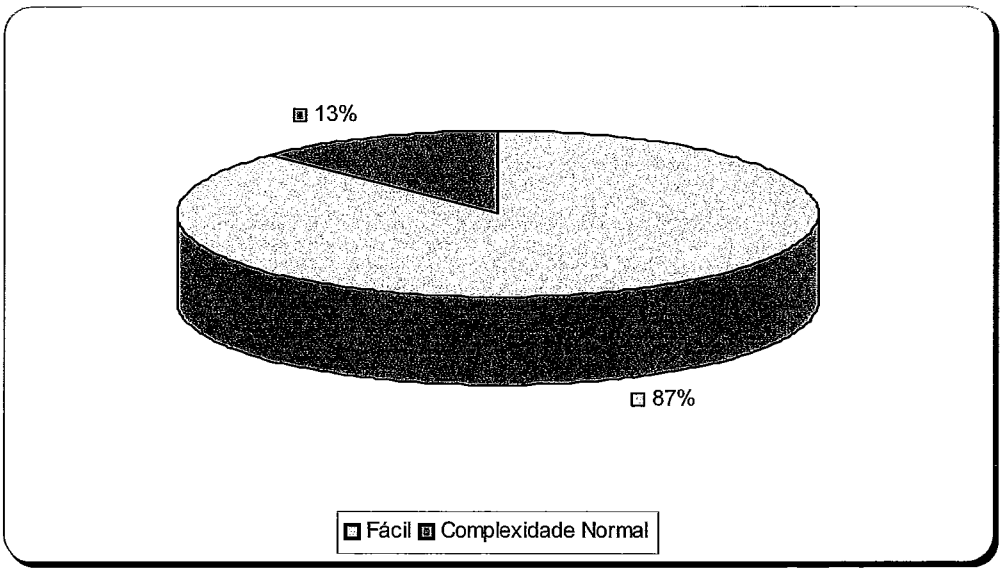


**Figura B.17 – Distribuição das respostas quanto à facilidade de uso (estudo de caso)**

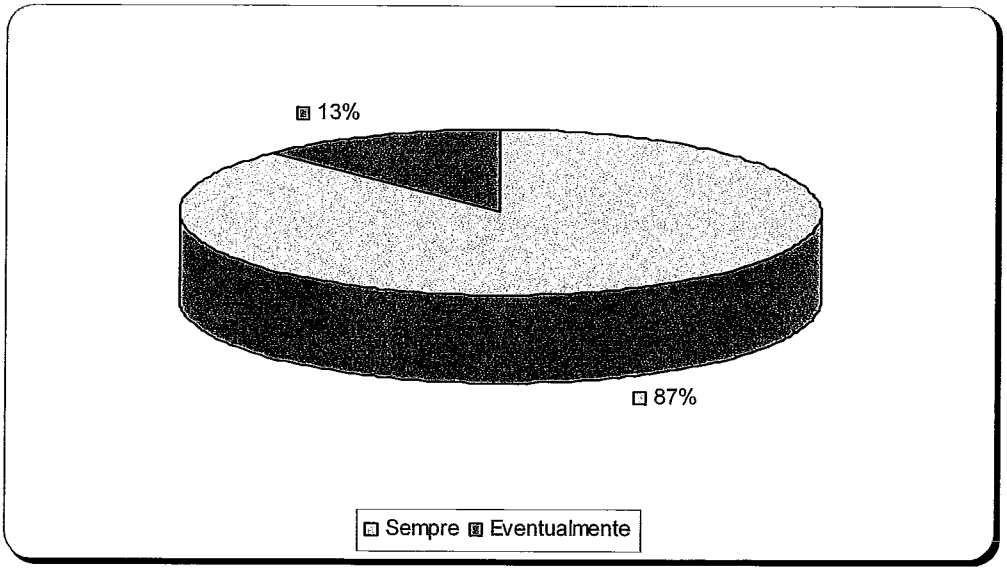


**Figura B.18 – Distribuição das respostas quanto à aderência da proposta aos seus objetivos iniciais (estudo de caso)**

# Anexo E

## Roteiro das Entrevistas de Validação da Ontologia

| Agrupamento | Construto em questão | Meta-Propriedade a ser avaliada | Pergunta ao Especialista |
|---|---|---|---|
| Fases do Processo de Mineração | Relação de Composição (Passo KDT -> Pré-Processamento) | Unidade | Pré-Processamento é SEMPRE uma fase do Processo de Text Mining? |
| | Relação de Composição (Passo KDT ->Mineração) | Unidade | Mineração é SEMPRE uma fase do Processo de Text Mining? |
| | Relação de Composição (Passo KDT -> Pós-Processamento) | Unidade | Pós-Processamento é SEMPRE uma fase do Processo de Text Mining? |
| | Classe | Identidade | Existe alguma fase do Processo de Text Mining que não seja Pré-Processamento, Mineração e Pós-Processamento? |
| | Classe | Identidade | Existe alguma outra fase do Processo de KDT que tenha a mesma função do Pré-Processamento? |
| | Classe | Identidade | Existe alguma outra fase do Processo de KDT que tenha a mesma função da Mineração? |
| | Classe | Identidade | Existe alguma outra fase do Processo de KDT que tenha a mesma função do Pós-Processamento? |

| | | | |
|---|---|---|---|
| Pré-Processamento | Relação de Composição (Pré-Processamento -> Função de Stemming) | Unidade | Stemming é SEMPRE uma função de Pré-Processamento? |
| | Relação de Composição (Pré-Processamento -> Função de Counting) | Unidade | Counting é SEMPRE uma função de Pré-Processamento? |
| | Relação de Composição (Pré-Processamento -> Função de Cleaning) | Unidade | Cleaning é SEMPRE uma função de Pré-Processamento? |
| | Classe | Identidade | Existe alguma função de Pré-Processamento que não foi considerada? |
| | Classe | Identidade | Existe alguma função de Pré-Processamento que tenha a mesma função de stemming? |
| | Classe | Identidade | Existe alguma função de Pré-Processamento que tenha a mesma função de Cleaning? |
| | Classe | Identidade | Existe alguma função de Pré-Processamento que tenha a mesma função de Counting? |
| Mineração de Texto | Relação de Composição (Mineração -> Classificação) | Unidade | Classificação é SEMPRE uma Tarefa de Mineração? |
| | Relação de Composição (Mineração -> Clusterização) | Unidade | Clusterização é SEMPRE uma Tarefa de Mineração? |
| | Classe | Identidade | Existe alguma tarefa de Mineração que não tenha sido considerada? |

| | | | |
|---|---|---|---|
| | Classe | Identidade | Existe alguma tarefa de mineração que tenha a mesma função de Classificação? |
| | Classe | Identidade | Existe alguma tarefa de mineração que tenha a mesma função de Clusterização? |
| Pós-Processamento | Relação de Composição (Pós Processamento -> Visualização) | Unidade | Visualização é SEMPRE uma função de Pós-Processamento? |
| | Classe | Identidade | Existe algumaFunção de Pós-Processamento que não tenha sido considerada? |
| | Classe | Identidade | Existe alguma função de Pré-Processamento que tenha a mesma função de Visualização? |
| Métodos | Relação de Composição (Método -> Baseado em Árvores) | Unidade | Método Baseado em árvores é sempre um método de Mineração de textos? |
| | Relação de Composição (Método -> Baseado em Redes Neurais) | Unidade | Método Baseado em redes neurais é sempre um método de Mineração de textos? |
| | Relação de Composição (Método -> Baseado em Lógica Fuzzy) | Unidade | Método Baseado em lógica fuzzy é sempre um método de Mineração de textos? |
| | Classe | Unidade | É possivel que algum método dos relacionados não seja um metodo de TM? |
| Software | Relação de Composição (Software -> Software Pré-Processamento) | Unidade | Software Pré-Processamento é SEMPRE um Software de Text Mining? |

| | | | |
|---|---|---|---|
| | Relação de Composição (Software Passo KDT - >Software Mineração) | Unidade | Software Mineração é SEMPRE um Software de Text Mining? |
| | Relação de Composição (Software Passo KDT -> Software Pós-Processamento) | Unidade | Software Pós-Processamento é SEMPRE um Software de Text Mining? |
| | Classe | Identidade | Existe algum Software de Text Mining que não seja Software Pré-Processamento, Software Mineração e Software Pós-Processamento? |
| | Classe | Identidade | Existe algum outro Software de KDT que tenha a mesma função do Software Pré-Processamento? |
| | Classe | Identidade | Existe alguma outro Software de KDT que tenha a mesma função da Software Mineração? |
| | Classe | Identidade | Existe alguma outro Software de KDT que tenha a mesma função do Software Pós-Processamento? |
| Software de Pré-Processamento | Relação de Composição (Software Pré-Processamento -> Software de Stemming) | Unidade | Software Stemming é SEMPRE um Software de Pré-Processamento? |
| | Relação de Composição (Software Pré-Processamento -> Software de Counting) | Unidade | Software Counting é SEMPRE um Software de Pré-Processamento? |
| | Relação de Composição (Software Pré-Processamento -> Software de Cleaning) | Unidade | Software Cleaning é SEMPRE um Software de Pré-Processamento? |
| | Classe | Identidade | Existe algum Software de Pré-Processamento que não foi considerado? |

| | | | |
|---|---|---|---|
| | Classe | Identidade | Existe algum Software de Pré-Processamento que tenha a mesma função de stemming? |
| | Classe | Identidade | Existe algum Software de Pré-Processamento que tenha a mesma função de Cleaning? |
| | Classe | Identidade | Existe algum Software de Pré-Processamento que tenha a mesma função de Counting? |
| Software de Mineração de Texto | Relação de Composição (Software Mineração -> Software Classificação) | Unidade | Software Classificação é SEMPRE um Software de Mineração? |
| | Relação de Composição (Software Mineração -> Software Clusterização) | Unidade | Software Clusterização é SEMPRE um Software de Mineração? |
| | Classe | Identidade | Existe algum Software de Mineração que não tenha sido considerado? |
| | Classe | Identidade | Existe algum Software de mineração que tenha a mesma função de Software Classificação? |
| | Classe | Identidade | Existe algum Software de mineração que tenha a mesma função de Software Clusterização? |
| Software de Visualização | Relação de Composição (Software Pós Processamento -> Software Visualização) | Unidade | Software Visualização é SEMPRE um Software de Pós-Processamento? |
| | Classe | Identidade | Existe algum Software de Pós-Processamento que não tenha sido considerado? |

| | Classe | Identidade | Existe algum Software de Pré-Processamento que tenha a mesma função de Software Visualização? |
|---|---|---|---|
| Propriedades das Classes/Instâncias | Propriedade CLUSTERS | Essencialidade | Todo Software engloba pelo menos um passo do processo de KDT ou pelo menos uma função/tarefa do processo? |
| | Propriedade PRECEEDS | Essencialidade | Uma fase do processo de KDT sempre precede uma outra na ordem sequencial de execução das fases? |
| | Propriedade BASED_ON | Essencialidade | Um algoritmo SEMPRE está baseado em um método proposto? |
| | Propriedade PERFORMS | Essencialidade | Um Software SEMPRE executa uma função/tarefa do processo de mineração de Textos? |
| | Propriedade IMPLEMENTS_OF | Essencialidade | Um software SEMPRE implementa um algoritmo que foi proposto? |
| | Propriedade SPECIFIES | Essencialidade | Um método sempre especifica uma fase do processo de KDT? |
| | Propriedade TYPE | Essencialidade | Todo software possui um Tipo? |
| | Propriedade INSTITUTION | Essencialidade | Todo software possui uma instituição? |
| | Propriedade DATA_TYPE | Essencialidade | Todo software possui um tipo de dado associado? |

| | | | |
|---|---|---|---|
| | Propriedade NAME | Essencialidade | Todo software possui um Nome? |
| | Propriedade COMENTARY | Essencialidade | Todo software possui um comentário? |
| | Propriedade AUTHOR | Essencialidade | Todo software possui um autor? |
| | Propriedade VERSION | Essencialidade | Todo software possui uma versão? |
| | Propriedade OBJECTIVE | Essencialidade | Todo software possui um objetivo? |
| | Propriedade AVAILABLE | Essencialidade | Todo software está disponível ou não? |
| | Propriedade WSDL_PATH | Essencialidade | Todos os softwares possuirão um WSDL associado? |